STAT40810 — Stochastic Models

Brendan Murphy

Week 3

Wrong Model

What happens if we fit the wrong model?

- When constructing a statistical model, we usually make a number of assumptions.
- One may ask "What happens if we fit the wrong model?"
- We would like to feel that inferences are somewhat robust to modeling choice.
- I will demonstrate some weak form of robustness to model choice.
- This offers a modicum of comfort when choosing a model.

Preliminary: Kullback-Leibler Divergence

- A useful measure of difference between two probability distributions is Kullback-Leibler divergence.
- It is defined as follows:

$$D(f||g) = \int f(x) \log \frac{f(x)}{g(x)} dx \text{ when continuous}$$
$$D(f||g) = \sum f(x) \log \frac{f(x)}{g(x)} dx \text{ when discrete}$$

- It is worth noting that:
  - $D(f||g) \geq 0$
  - $D(f||g) = 0$ if and only if $f = g$.

Maximum Likelihood

- Suppose that we have $x_1, x_2, \ldots, x_n$ which are sampled from some model $g(x)$.
- But we model the data as coming from $f(x|\theta)$ with unknown θ.
- Let's assume that we have continuous values (for simplicity)
- When fitting a model, we may use maximum likelihood.
- When doing this we find a value of θ that maximizes

$$\ell(\theta) = \sum_{i=1}^{n} \log f(x_i|\theta).$$

- Equivalently, we get a θ that minimizes

$$-\ell(\theta) = -\sum_{i=1}^{n} \log f(x_i|\theta).$$

Maximum Likelihood 2

- Equivalently, we get a θ that minimizes

$$-\frac{\ell(\theta)}{n} = -\frac{1}{n}\sum_{i=1}^{n} \log f(x_i|\theta).$$

- As $n \to \infty$, we get

$$-\frac{1}{n}\sum_{i=1}^{n} \log f(x_i|\theta) \to -\mathbb{E}\log f(x|\theta) = -\int g(x)\log f(x|\theta)dx$$

- So, as $n \to \infty$ we are minimizing

$$-\int g(x)\log f(x|\theta)dx$$

with respect to θ.

- This is equivalent to minimizing

$$\int g(x)\log g(x)dx - \int g(x)\log f(x|\theta)dx.$$

Maximum Likelihood 3

- So, we are effectively minimizing

$$\begin{aligned} D(g||f(x|\theta)) &= \int g(x)\log g(x)dx - \int g(x)\log f(x|\theta)dx \\ &= \int g(x)\log\frac{g(x)}{f(x|\theta)}dx \end{aligned}$$

- That is, maximum likelihood is (approximately) finding the closest model to the correct one.

STAT40810 — Stochastic Models

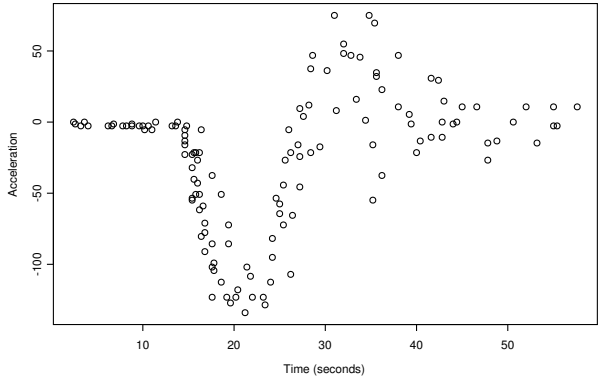
Brendan Murphy

Week 3

Nonparametric Regression

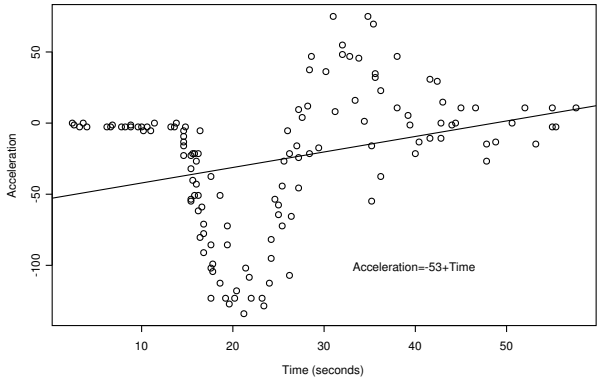
Example: Motorcycle Crash Simulation

- Data were collected recording the acceleration versus time for a simulated motorcycle accident. Here is a scatter plot of the data.



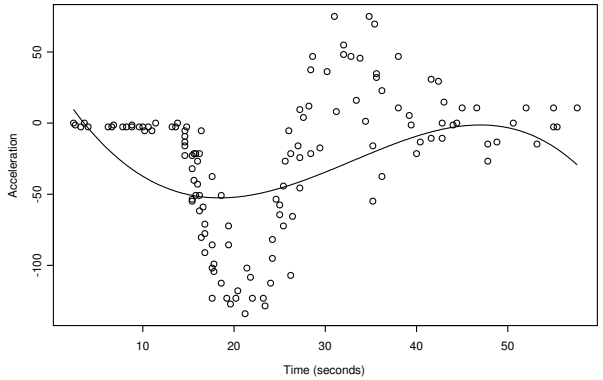
Linear Regression

- A line was fitted to the data using least squares and we got the following fit:



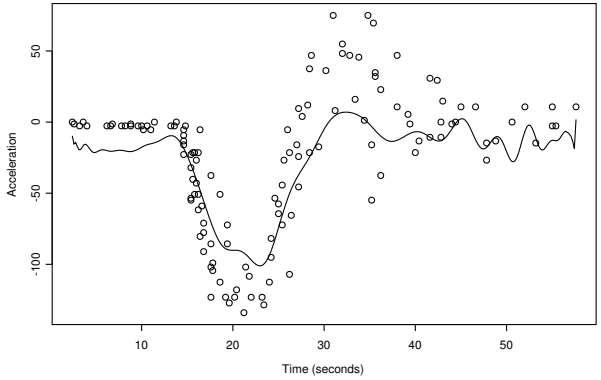
Polynomial Regression

- A quartic was fitted to the data using least squares and we got the following fit:



Polynomial Regression

- A polynomial of degree 40 was fitted to the data using least squares and we got the following fit:



Comments and Potential Problems

- We can immediately see that the linear regression fails miserably for this data.
- The polynomial regression appears to do much better, provided that we use a high enough order polynomial.
- We could use a polynomial of degree 132 (there were 133 data points) and we would get a perfect fit.
- There are numerical problems with fitting high order polynomials because we need to invert a matrix with very small determinant.
- The numerical problems can be reduced if we use orthogonal polynomials, for example, Hermite polynomials.
- Polynomial regression can give terrible results when we extrapolate.

Non-parametric Regression

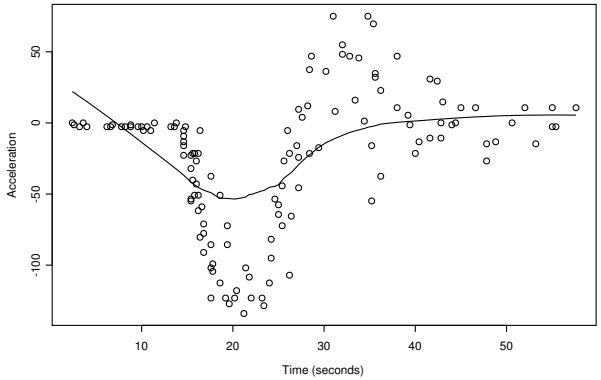
- Non-parametric regression includes a vast number of methods which are used to fit relationships while making few assumptions.
- Non-parametric regression is commonly used to smooth data, making it less noisy than the original data.
- Methods which we will describe are:
  - Lowess (Locally Weighted Regression)
  - Kernel Smoothing
  - Smoothing Splines
- Parts of these names may be familiar.

Lowess Regression

- Lowess regression is a variant on linear regression, where it does a different regression to predict each point.
- The regression used to predict each data point uses only a fraction of the data (the fraction is usually called the *span*).
- The regression also weights the observations according to how far they are away from the point being predicted.
- This will be described in more detail later.

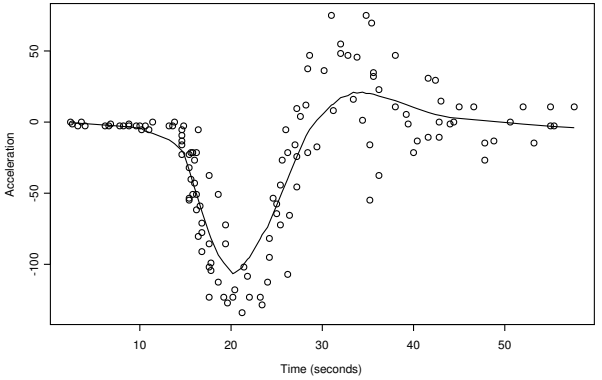
Lowess Regression Results

- A lowess regression was completed and the following curve was fitted:



Lowess Regression Results II

- Another lowess regression was completed with a different smoothing parameter and the following curve was fitted:



Kernel Smoothing

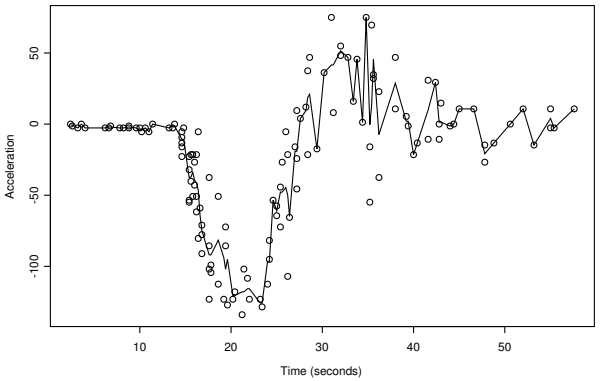
- Kernel smoothing is an alternative method of using weights to get a good local fit.
- The method requires specifying a kernel and bandwidth parameter. The choice of bandwidth is very important, whereas the choice of kernel is less important.
- The value of the response variable is predicted using $\hat{y}_j = \sum_{i=1}^{n} w_{ij} y_i$ where

$$w_{ij} = \frac{\frac{1}{h} K\left(\frac{x_i - x_j}{h}\right)}{\frac{1}{h}\sum_{k=1}^{n} K\left(\frac{x_i - x_k}{h}\right)}$$

- More on this later...

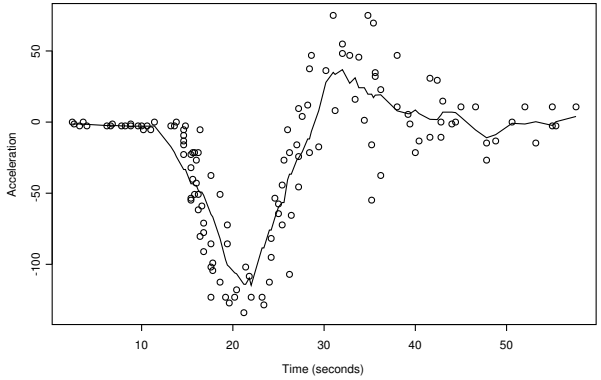
Kernel Smoothing Results

- The kernel smoother with a rectangular kernel and bandwidth 0.5 gave the following fitted curve.



Kernel Smoothing Results

- The kernel smoother with a rectangular kernel and bandwidth 5 gave the following fitted curve.



Smoothing Splines

- We could find the function, $f(x)$, that minimizes

$$\sum_{j=1}^{n} [y_j - f(x_j)]^2,$$

but the resulting function would just interpolate the points (x_j, y_j).

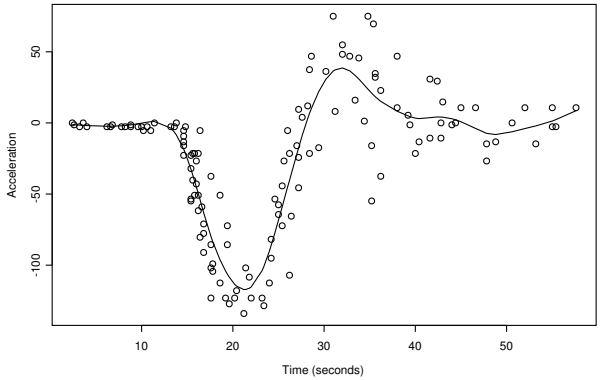
- The penalised sum of squares criterion chooses a function $f(x)$ which minimizes

$$\sum_{j=1}^{n} [y_j - f(x_j)]^2 + \lambda \int_{\mathcal{X}} f''(x)^2 dx, \text{ where } \lambda > 0.$$

- The λ value controls how smooth we want $f(x)$ to be.
- $\lambda = 0$ would give a function that interpolates (x_j, y_j) whereas $\lambda = \infty$ would give a linear function.

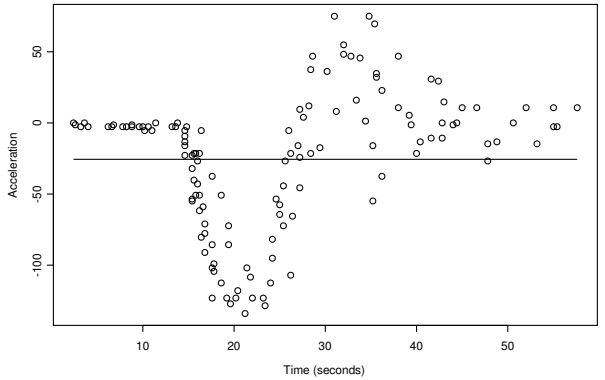
Smoothing Spline Results

- The smoothing spline when the smoothing parameter was chosen by cross-validation.



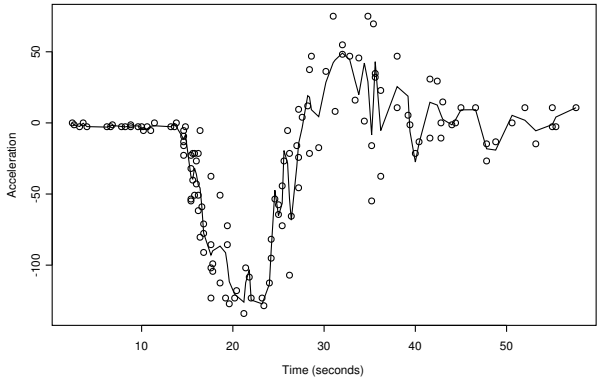
Smoothing Spline Results

- The smoothing spline when the smoothing parameter was very large.



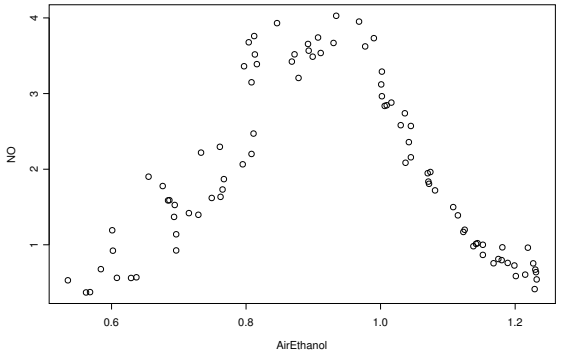
Smoothing Spline Results

- The smoothing spline when the smoothing parameter was very small.



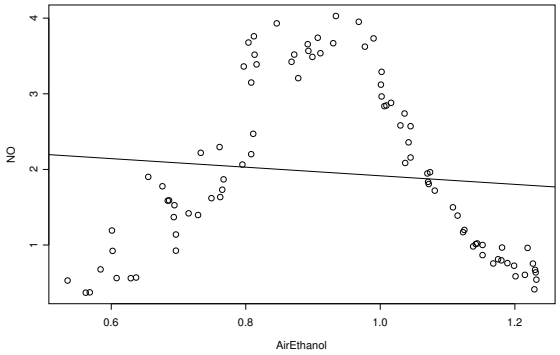
Example: Nitric Oxide Emissions

- Data were collected recording the amount of nitric oxide (NO) emitted from engines, where the air/ethanol mix of the fuel was varied.
- The relationship between the air/ethanol mix and the nitric oxide emissions was of interest.



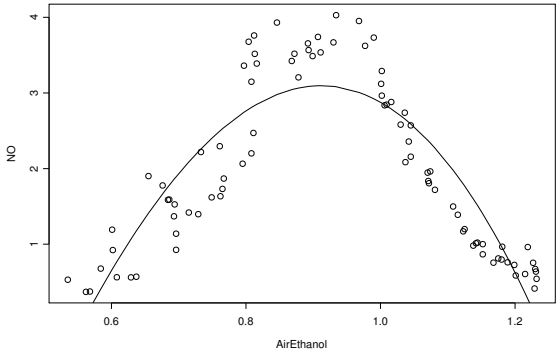
Linear Regression

A line was fitted, but the fitted relationship was very poor.



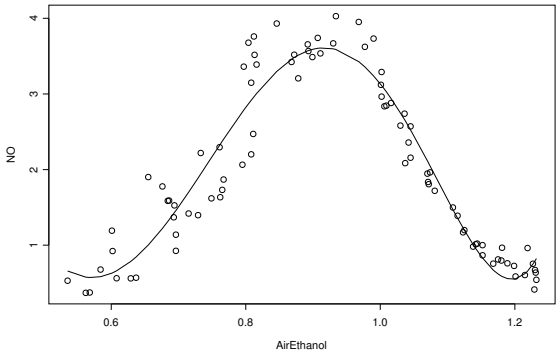
Cubic Regression

A cubic equation was fitted, but the fitted relationship was (again!) very poor.



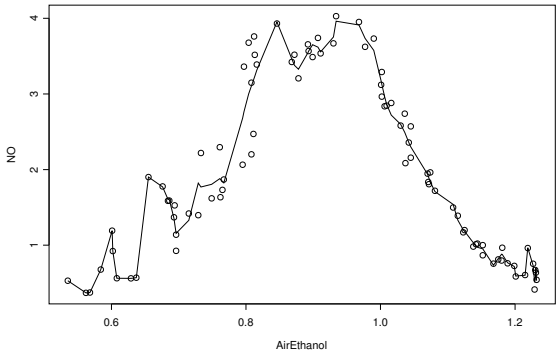
Quintic Regression

A quintic equation was fitted, but the fitted relationship is getting better, but gives poor extrapolations.



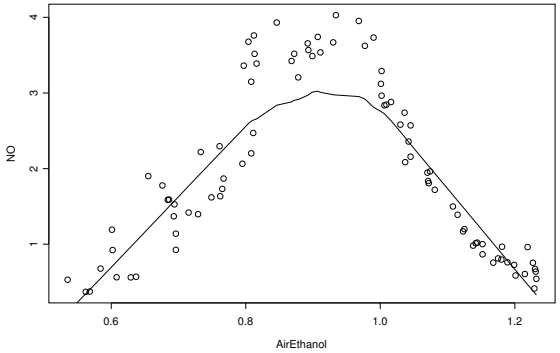
Polynomial Regression

A degree 50 polynomial equation was fitted, but the fitted relationship is getting better, but we are definitely overfitting.



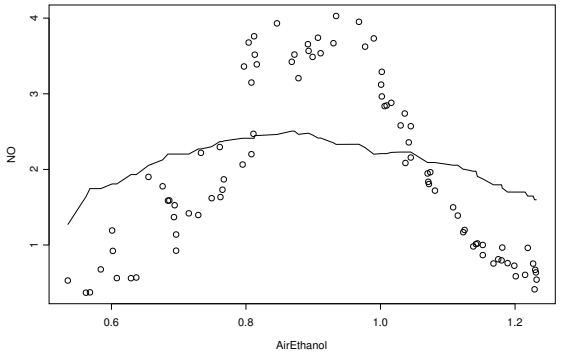
Lowess

A LOWESS regression was completed. The LOWESS curve finds it hard to adjust to the changing curvature.



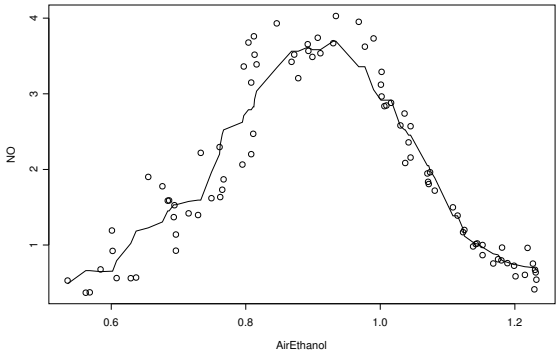
Kernel Smoothing

A kernel smoother was used. The results are very poor.



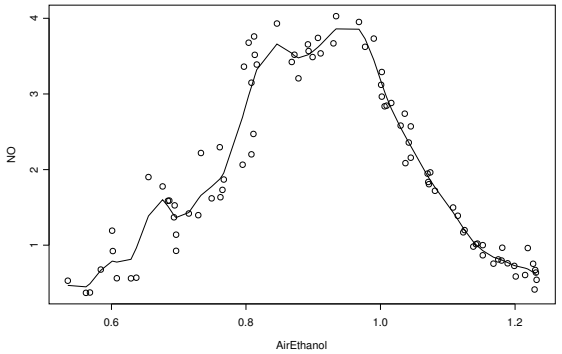
Kernel Smoothing

A kernel smoother was used, with a narrower bandwidth. The results much better.



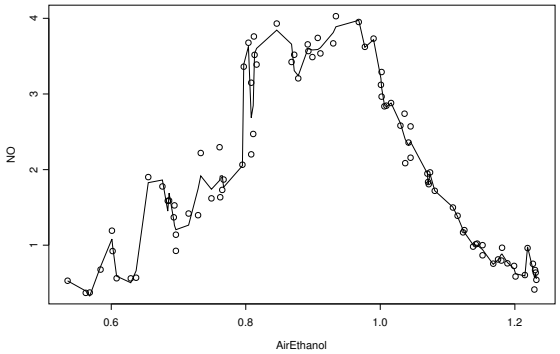
Splines

A spline regression was completed. The fitted curve seems to find the ideal balance between fit and overfitting.



Splines

A spline regression was completed, with small roughness penalty. The fitted curve almost interpolates the data!



STAT40810 — Stochastic Models

Brendan Murphy

Week 3

Locally Weighted Regression (LOWESS)

Locally Weighted Regression Smoothing (LOWESS)

- Suppose we have a dataset with values $(x_1, y_1), \ldots, (x_n, y_n)$.
- We want to study the relationship between the x and the y values.
- In other words, we want to estimate the function $f(x_i)$, where $y_i = f(x_i) + \epsilon_i$.
- A crucial input into LOWESS is the the *span* (s).
  The span determines what proportion of the data is used when estimating $f(x)$ near a point $x = x_i$.
- The span is a number between 0 and 1; it controls the smoothness of the estimate $\hat{f}(x)$.

LOWESS (I)

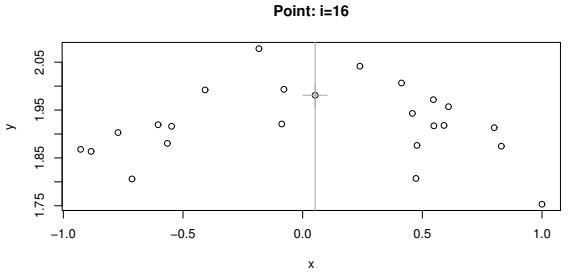
- To estimate the value of f at the point x_i, that is $\hat{f}(x_i)$ we take the following steps:
  1. Compute the value of $k = \lceil sn \rceil$.
  2. Find the k nearest neighbors of x_i. These points constitute a neighborhood $N(x_i)$ of x_i.
  3. Calculate the largest distance between x_i and another point in the neighborhood:
  $$\Delta(x_i) = \max_{x \in N(x_i)} |x_i - x|.$$

LOWESS (II)

4. Assign weights to each point in $N(x_i)$ using the tri-cube weight function:
$$w\left(\frac{|x - x_i|}{\Delta(x_i)}\right)$$
where
$$w(u) = \begin{cases} (1 - |u|^3)^3 & \text{for } |u| \leq 1 \\ 0 & \text{otherwise} \end{cases}.$$
5. Calculate the weighted least squares fitting line on the data in the neighborhood $N(x_i)$.
6. Let $\hat{f}(x_i) = \hat{y}_i = \hat{a} + \hat{b}x_i$.

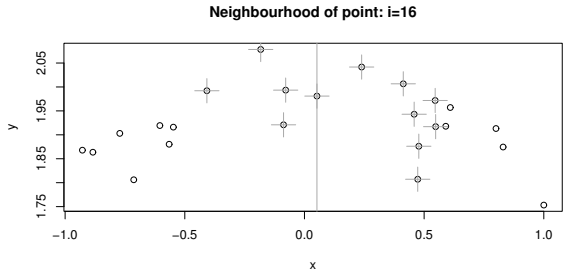
LOWESS Steps (1)

- Suppose we want to compute $\hat{f}(x_i)$. Suppose $s = 0.5$ and $n = 24$. Hence, $k = 12$.



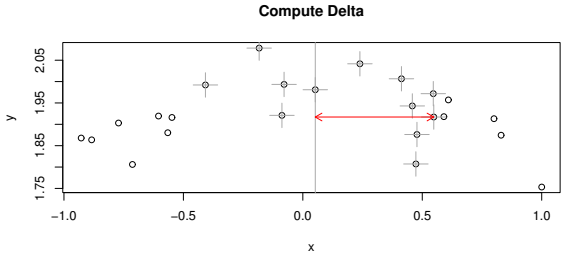
LOWESS Steps (2)

- Compute the neighbourhood $N(x_i)$.



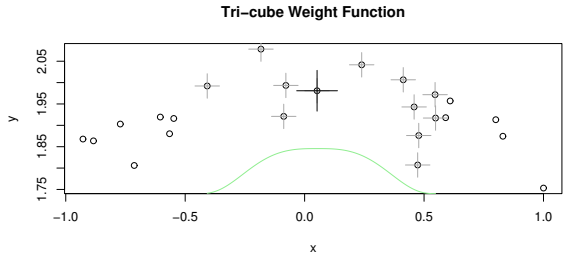
LOWESS Steps (3)

- Compute the value of Δ



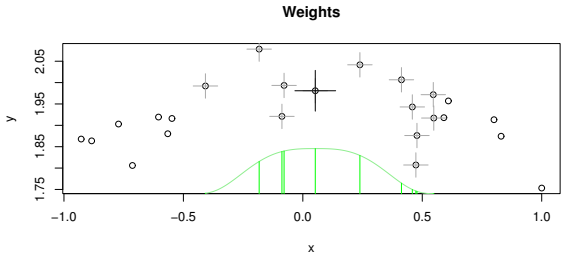
LOWESS Steps (4a)

- The tri-cube weight function is computed.



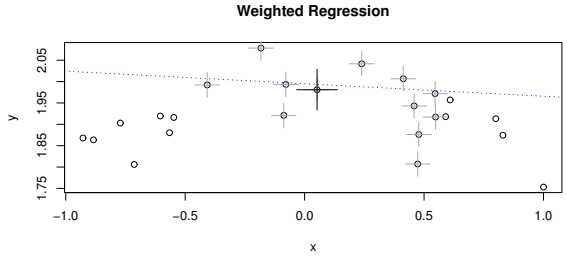
LOWESS Steps (4b)

- The weights are computed.



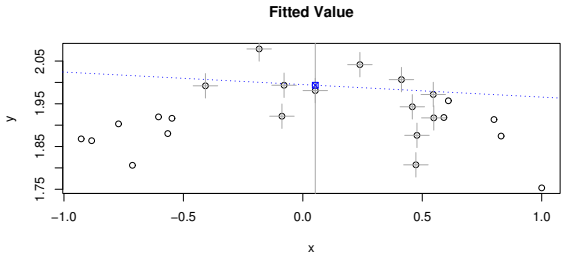
LOWESS Steps (5)

- A weighted least squares fit is found.



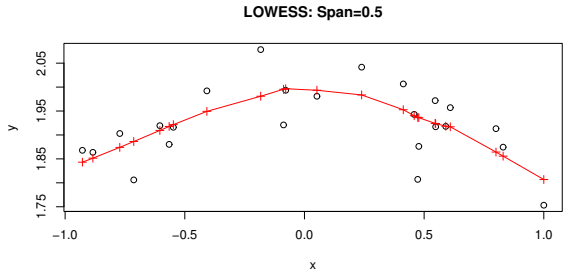
LOWESS Steps (6)

- The fitted value is found.



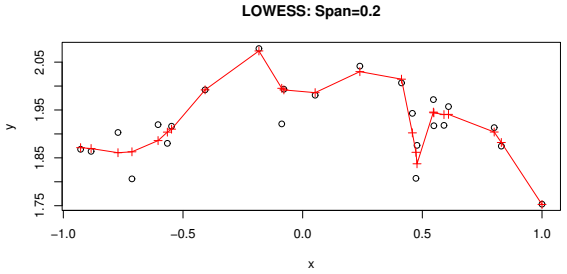
LOWESS Fits

- The steps are repeated for each $i = 1, 2, \ldots, n$. This gives the LOWESS curve.



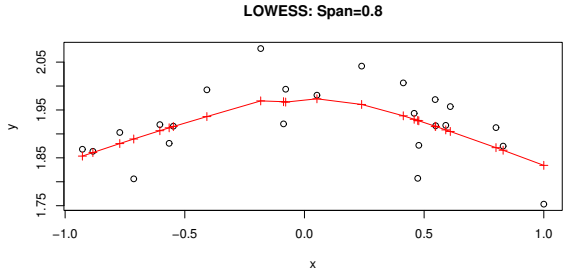
LOWESS (Span=0.2)

- If the span is smaller, then we get a more "wiggly" fit.



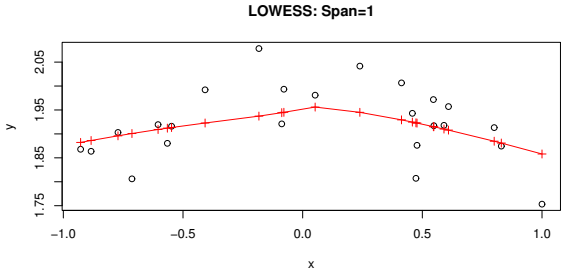
LOWESS Fits (Span=0.8)

- If the span is larger, then we get a more "smooth" fit.



LOWESS Fits (Span=1.0)

- If the span is even larger, then we get an even "smoother" fit.



LOWESS

- LOWESS is very easy to implement in R.
- Here's code to model the motorcycle data.

```
# Load the data
library(MASS)
data(mcycle)

# Plot the data
plot(mcycle)

# Fit a lowess regression
# The value of f controls the span
fit <- lowess(mcycle$times,mcycle$accel,f=2/3)

# Add the fitted values to the plot
points(fit,pch=3,col="red")

# Assess fit using mean squared error (MSE)
MSE <- mean((mcycle$accel-fit$y)^2)
MSE
```


STAT40810 — Stochastic Models

Brendan Murphy

Week 3

Kernel Smoothing

Kernel Smoothing

- Suppose we have a dataset with values $(x_1, y_1), \ldots, (x_n, y_n)$.
- We want to study the relationship between the x and the y values.
- In other words, we want to estimate the function $f(x_i)$, where $y_i = f(x_i) + \epsilon_i$.
- A crucial input into kernel smoothing is the kenrel and the *bandwidth* (h).
- The bandwidth is a number greater than 0; it controls the smoothness of the estimate $\hat{f}(x)$.

Kernel Function

- A kernel is a function $K(x)$ such that

$$K(x) > 0 \text{ and } \int_{\mathcal{X}} K(x)dx = 1.$$

- Most kernels are also symmetric, so $K(x) = K(-x)$.
- Some commonly used kernels are:

$$K(x) = \frac{1}{\sqrt{2\pi}} \exp\left(-\frac{x^2}{2}\right) \quad \text{Gaussian Kernel}$$

$$K(x) = \begin{cases} 1 - |x| & \text{for } |x| < 1 \\ 0 & \text{otherwise} \end{cases} \quad \text{Triangular Kernel}$$

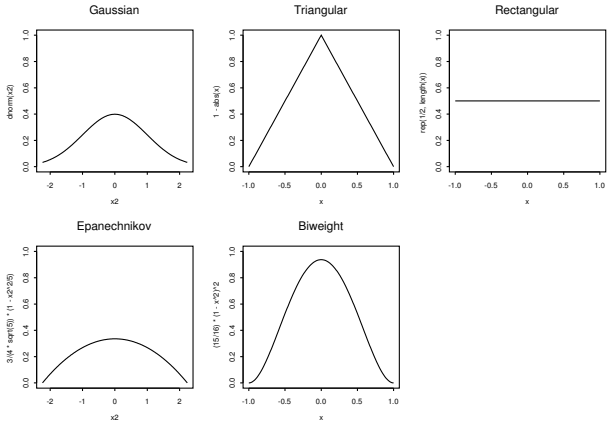
Yet More Kernels

$$K(x) = \begin{cases} \frac{1}{2} & \text{for } |x| < 1 \\ 0 & \text{otherwise} \end{cases} \qquad \text{Rectangular Kernel}$$

$$K(x) = \begin{cases} \frac{3}{4\sqrt{5}}\left(1 - \frac{x^2}{5}\right) & \text{for } |x| < \sqrt{5} \\ 0 & \text{otherwise} \end{cases} \qquad \text{Epanechnikov Kernel}$$

$$K(x) = \begin{cases} \frac{15}{16}\left(1 - x^2\right)^2 & \text{for } |x| < 1 \\ 0 & \text{otherwise} \end{cases} \qquad \text{Biweight Kernel}$$

Various Kernels

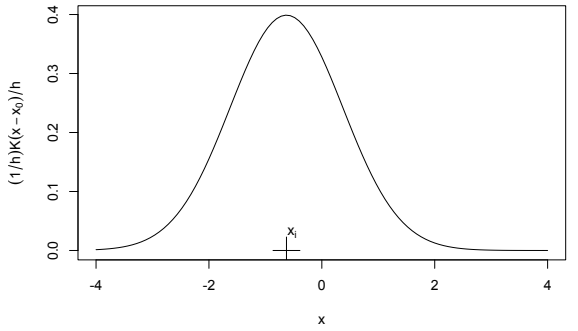


Bandwidth

- If we take a kernel function $K(x)$, any x_0 and a bandwidth $h > 0$ and compute
$$\frac{1}{h}K\left(\frac{x - x_0}{h}\right).$$
- This has the effect of shifting the kernel to being centred on x_0 instead of 0.
- It also has the effect of:
  - Making the kernel narrower and taller, if $h < 1$.
  - Making the kernel wider and lower, if $h > 1$.
- The value of h is called the bandwidth.

Kernel Plot

- An example of what
$$\frac{1}{h}K\left(\frac{x - x_0}{h}\right)$$
looks like is:



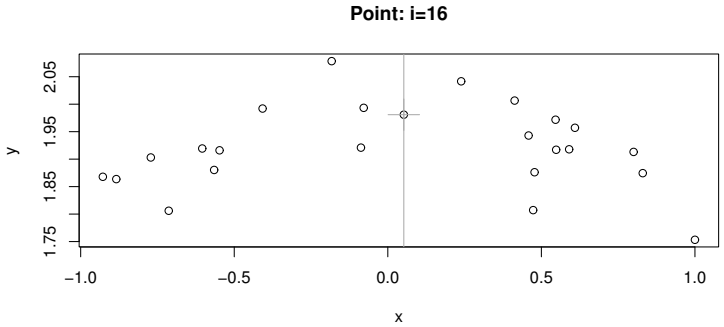
Kernel Smoothing

- Kernel smoothing requires us to specify a kernel and a bandwidth.
- The value of the response variable is predicted using $\hat{f}(x_i) = \hat{y}_i = \sum_{j=1}^{n} w_{ij} y_j$ where

$$w_{ij} = \frac{\frac{1}{h} K\left(\frac{x_i - x_j}{h}\right)}{\frac{1}{h} \sum_{k=1}^{n} K\left(\frac{x_i - x_k}{h}\right)}$$

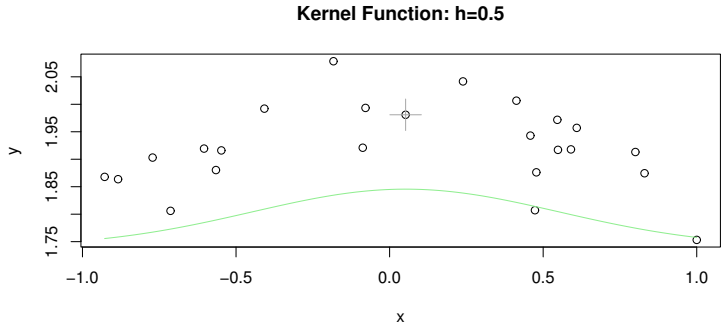
Kernel Smoothing Steps (1)

- Suppose we want to compute $\hat{f}(x_i)$. Suppose $h = 0.5$ and $n = 24$.



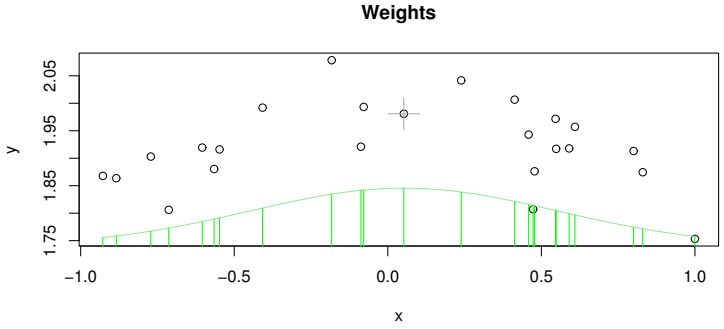
Kernel Smoothing Steps (2)

- The kernel function $(1/h)K\left(\frac{x - x_i}{h}\right)$ is computed. A Gaussian kernel is shown.



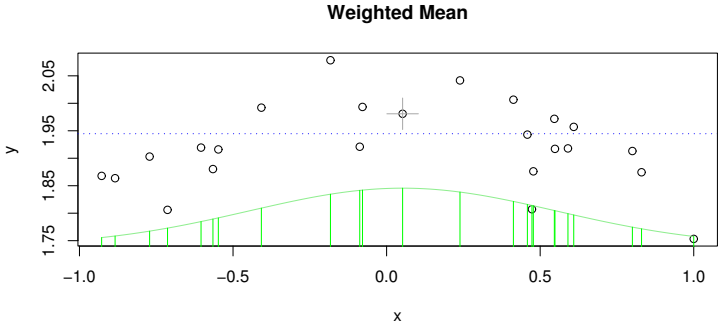
Kernel Smoothing Steps (3)

- The weights are computed.



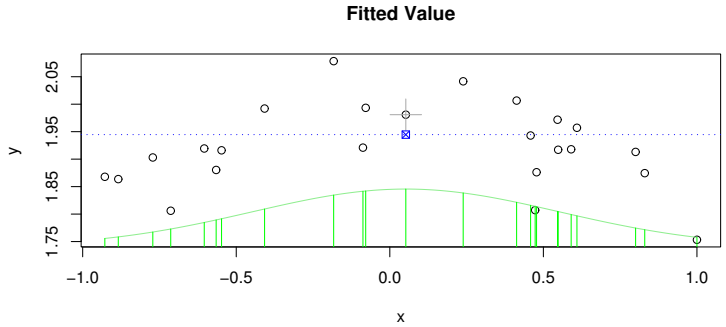
Kernel Smoothing Steps (4)

- A weighted mean is calculated



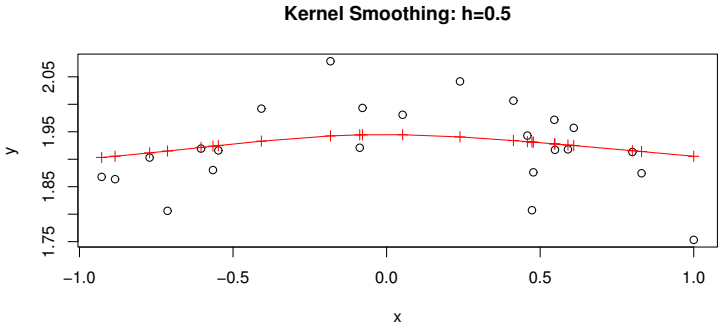
Kernel Smoothing Steps (5)

- The fitted value is found (it's just the weighted mean!)



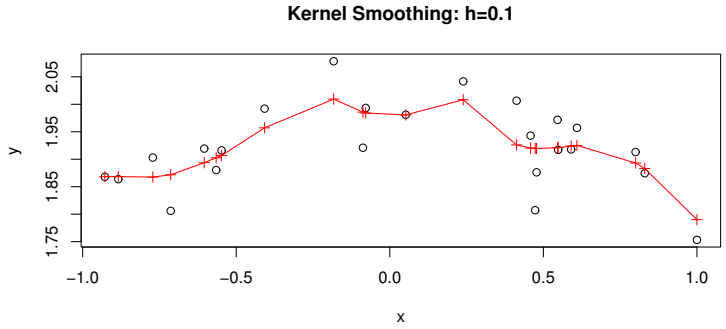
Kernel Smoothing Fits

- The steps are repeated for each $i = 1, 2, \ldots, n$. This gives the kernel smoothed curve.



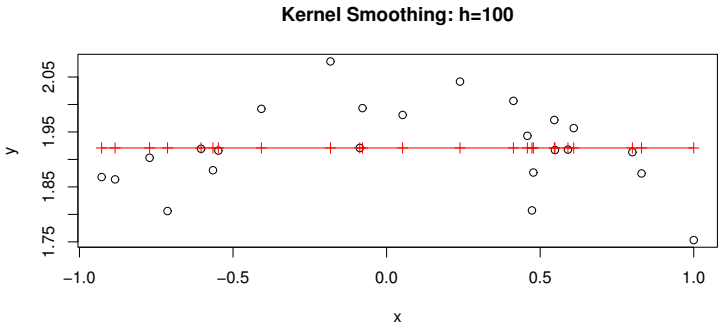
Kernel Smoothing (h=0.1)

- If the bandwidth is smaller, then we get a more "wiggly" fit.



Kernel Smoothing (h=100)

- If the span is very large, then we get a constant function, equal to the mean of the ys.



Variants

- Variants of kernel smoothing exist which use weighted polynomial regression instead of weighted means.
- Can you see the similarity with LOWESS?

Kernel Smoothing

- Kernel Smoothing is very easy to implement in R.
- Here's code to model the motorcycle data.

```
# Load the data
library(MASS)
data(mcycle)

# Plot the data
plot(mcycle)

# Fit a kernel smoothing
# The bandwidth needs to be specified
fit <- ksmooth(mcycle$times,mcycle$accel,kernel="normal",bandwidth=3)

# Add the fitted values to the plot
points(fit,pch=3,col="red")

# Assess fit using mean squared error (MSE)
MSE <- mean((mcycle$accel-fit$y)^2)
MSE
```


STAT40810 — Stochastic Models

Brendan Murphy

Week 3

Spline Regression

Splines

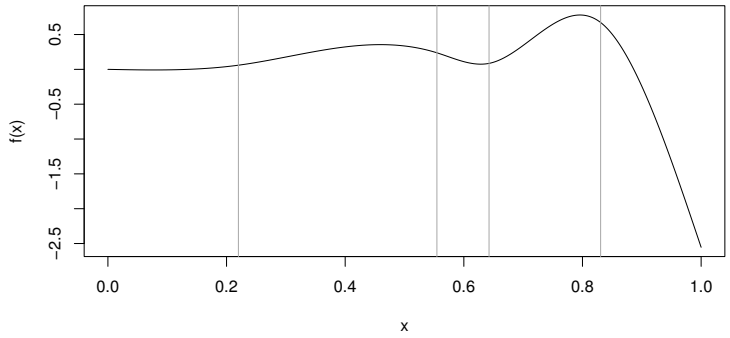
- We will look at spline regression as a precursor to doing smoothing splines. regression.
- First, we need to look at splines in general.
- **Potential Books Of Interest:**
  - Ramsay and Silverman (1997) Functional Data Analysis.
  - Hastie and Tibshirani (1990) Generalized Additive Models.
  - Hastie, Tibshirani and Friedman (2001) The Elements of Statistical Learning.
  - de Boor (1978) A Practical Guide to Splines.

Splines

- A Kth order spline with M knots located at $z_1 < z_2 < \cdots < z_M$ is a function $f(x)$ such that
  1. The function is a Kth order polynomial in the intervals (z_i, z_{i+1}).
  2. The function has $K-1$ continuous derivatives.
- A spline of order K with M knots has $(K+1)(M+1) - KM = K + M + 1$ parameters.
  The $(K+1)(M+1)$ term is for the equation between the knots and the KM term is for the boundary conditions at the knots.
- We will primarily concentrate on cubic splines.

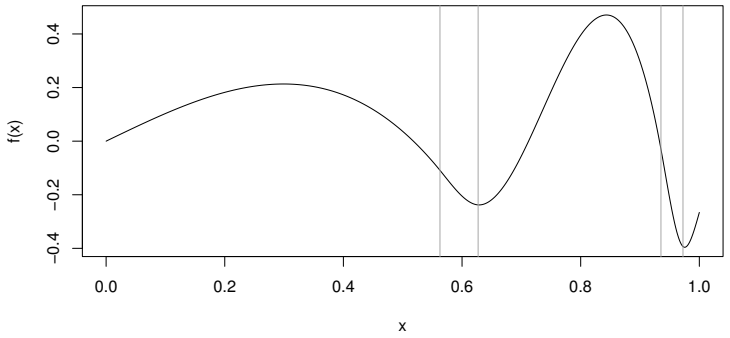
Example 1: Spline

- Here's an example of a cubic spline $f(x)$. The knots are marked in grey.



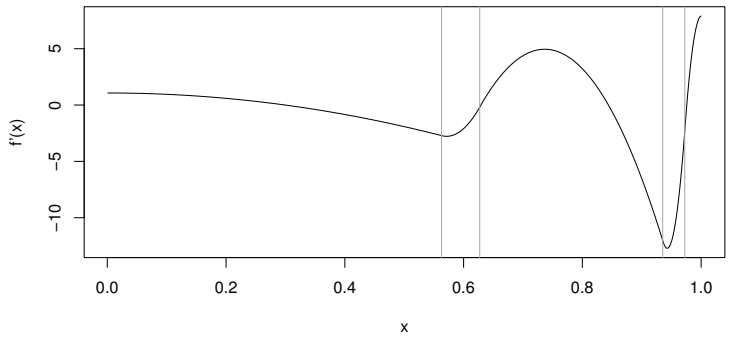
Example 2: Spline

- Here's another example of a cubic spline $f(x)$. The knots are marked in grey.



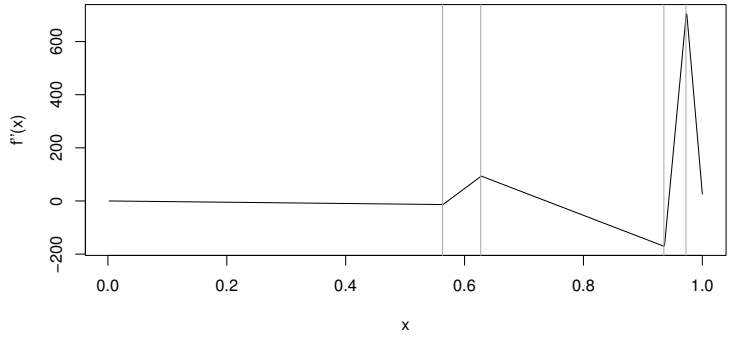
Example 2: Spline (First Derivative)

- The derivative $f'(x)$ of the spline is a spline of order 2. Notice the quadratic shape between knots.



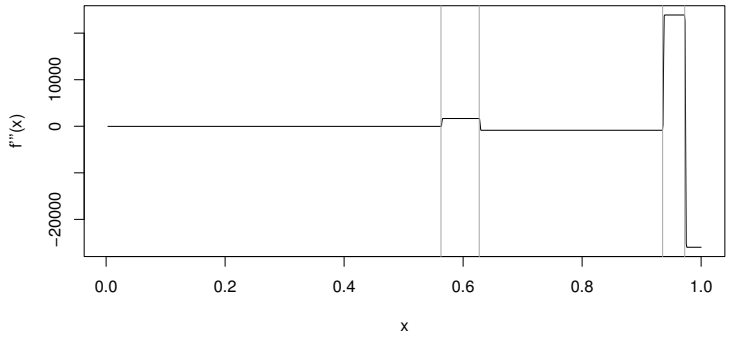
Example 2: Spline (Second Derivative)

- The second derivative $f''(x)$ of the spline is piecewise linear and continuous.



Example 2: Spline (Third Derivative)

- The third derivative is piecewise constant; that is, where it exists. It doesn't necessarily exist at the knots.



Spline Basis Functions

- A spline of order K with M knots at $z_1 < z_2 < \cdots < z_M$, can be written as a linear combination of basis functions.
- That is,

$$f(x) = b_0 + \sum_{l=1}^{M+K} b_l f_l(x).$$

- There are a number of ways of doing this.
- One standard way is to write

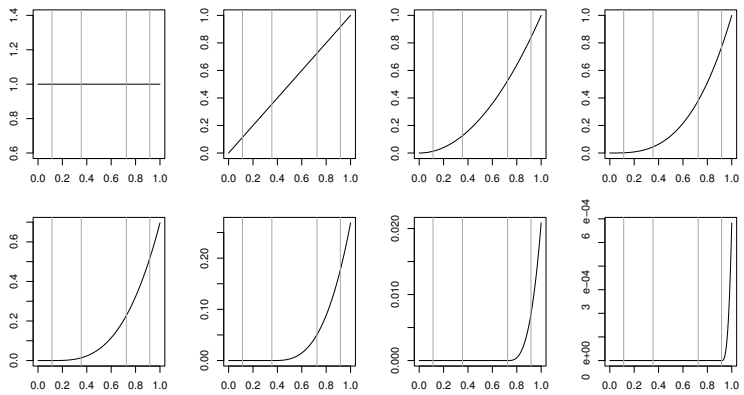
$$f(x) = b_0 + \sum_{k=1}^{K} b_k x^k + \sum_{m=1}^{M} b_{K+m}(x - z_k)_+^K,$$

where $(x - z_k)_+$ is the positive part of $(x - z_k)$.

- This is called the *truncated power series basis*.

Spline Basis Functions

- An example of the truncated power series basis is shown.



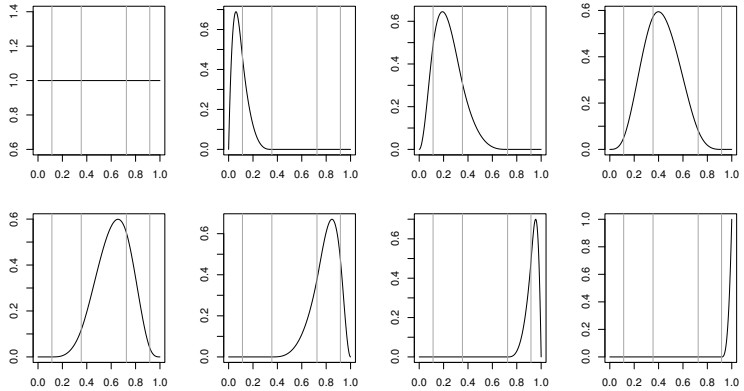
- This basis is not computationally efficient when the number of knots is large.

B-Spline Basis

- The truncated power series basis for splines is not suitable when the number of knots is large.
- An alternative basis called the *B-spline basis* offers a computational advantage.
- Each basis function in the *B*-spline basis has bounded support (ie. positive over a finite interval).
- This simplifies some calculations needed when using splines.

B-Spline Basis

- An example of the *B*-spline basis for the same knots as the previous truncated power series basis is shown.



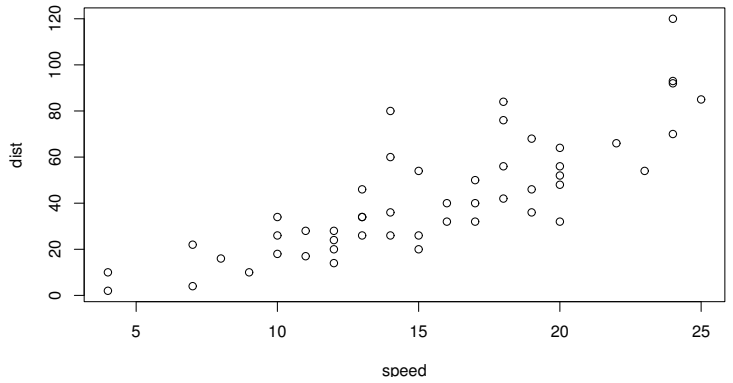
Spline Regression

- Suppose that we have data values $(x_1, y_1), (x_2, y_2), \ldots, (x_n, y_n)$.
- We could model the relationship between the variables using a spline function.
- If we specify the knot locations $z_1 < z_2 < \ldots < z_M$ and the order of the spline, then the problem is simply a regression problem.
- We can use regression techniques to find the coefficients $(b_0, b_1, \ldots, b_{K+M})$ that minimize

$$\sum_{i=1}^{n} \left[y_i - b_0 - \sum_{l=1}^{K+M} b_l f_l(x_i) \right]^2 .$$

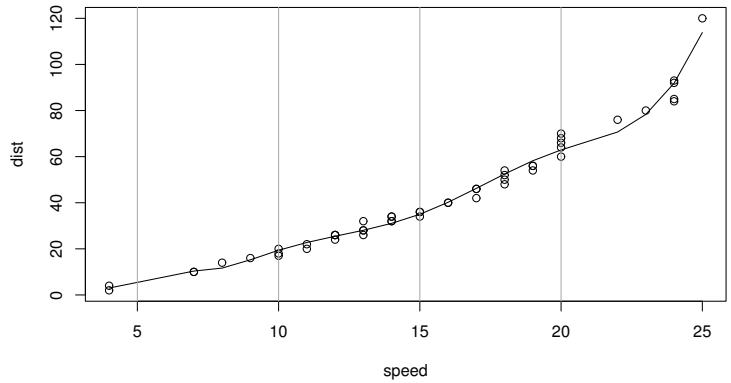
Example: Braking Distances

- The data were collected, recording the speed of cars and the distances taken to stop. Note that the data were recorded in the 1920s.
- There are 50 observations on 2 variables: Speed (mph) and Stopping Distance (ft).



Spline Regression

- Suppose we fit a cubic spline with knots at 10, 15 and 20ft.



Spline Regression

Here's code to model the motorcycle data.

```
#Load data and plot it
data(cars)
plot(cars)

# Load splines library
library(splines)

#Form B-spline basis functions for data
basis <- bs(cars$speed,knots=c(5,10,15,20))

# Fit the model
fit <- lm(cars$dist~basis)

#Show fit on the plot
plot(cars)
points(cars$speed,predict(fit),type="l",col="red")
```