Introduction to Regression

Chad M. Schafer
cschafer@stat.cmu.edu

Carnegie Mellon University
Outline

- General Concepts of Regression, Bias-Variance Tradeoff
- Linear Regression
- Nonparametric Procedures
- Cross Validation
- Local Polynomial Regression
- Confidence Bands
- Basis Methods: Splines
- Multiple Regression
Basic Concepts in Regression

The Regression Problem:

Observe \((X_1, Y_1), \ldots, (X_n, Y_n)\), estimate \(f(x) = \mathbb{E}(Y|X = x)\).

Equivalently: Estimate \(f(x)\) with

\[ Y_i = f(X_i) + \epsilon_i, \quad i = 1, 2, \ldots, n \]

where \(\mathbb{E}(\epsilon_i) = 0\).

Simple estimators:

\[
\hat{f}(x) = \hat{\beta}_0 + \hat{\beta}_1 x \quad \text{parametric}
\]

\[
\hat{f}(x) = \text{mean}\{Y_i : |X_i - x| \leq h\} \quad \text{nonparametric}
\]
Regression

Parametric Regression:

\[
Y = \beta_0 + \beta_1 X + \epsilon \\
Y = \beta_0 + \beta_1 X_1 + \cdots + \beta_d X_d + \epsilon \\
Y = \beta_0 e^{\beta_1 X_1} + \frac{\beta_2}{X_2} + \epsilon
\]

Nonparametric Regression:

\[
Y = f(X) + \epsilon \\
Y = f_1(X_1) + \cdots + f_d(X_d) + \epsilon \\
Y = f(X_1, X_2) + \epsilon
\]
Next... 

We will first quickly look at a simple example from astronomy.
Example: Type Ia Supernovae

From Riess, et al. (2007), 182 Type Ia Supernovae
Example: Type Ia Supernovae

Assumption:

Observed pairs \((z_i, Y_i)\) are realizations of

\[ Y_i = f(z_i) + \sigma_i \epsilon_i, \]

where the \(\epsilon_i\) are i.i.d. standard normal with mean zero.

Error standard deviations \(\sigma_i\) assumed known.

Objective: Estimate \(f(\cdot)\).
Example: Type Ia Supernovae

\( \Lambda \text{CDM, two parameter model:} \)

\[
f(z) = 5 \log_{10} \left( \frac{c(1 + z)}{H_0} \int_0^z \frac{du}{\sqrt{\Omega_m(1 + u)^3 + (1 - \Omega_m)}} \right) + 25
\]
Example: Type Ia Supernovae

Estimate where $H_0 = 72.76$ and $\Omega_m = 0.341$ (the MLE)
Example: Type Ia Supernovae

Fourth order polynomial fit.
Example: Type Ia Supernovae

Fifth order polynomial fit.
The previous two slides illustrate the fundamental challenge of constructing a regression model: How complex should it be?
The Bias–Variance Tradeoff

Must choose model to achieve balance between

too simple – high bias, i.e., $\mathbb{E}(\hat{f}(x))$ not close to $f(x)$
precise, but not accurate

too complex – high variance, i.e., $\text{Var}(\hat{f}(x))$ large
accurate, but not precise

This is the classic bias–variance tradeoff.

Could be called the accuracy–precision tradeoff.
The Bias–Variance Tradeoff

Linear Regression Model:

Fit models of the form

$$\hat{f}(x) = \hat{\beta}_0 + \sum_{i=1}^{p} \hat{\beta}_i g_i(x)$$

where $g_i(\cdot)$ are fixed, specified functions.

Increasing $p$ yields more complex model
Linear regression models are the classic approach to regression. Here, the response is modelled as a linear combination of $p$ selected predictors.
Linear Regression

Assume that

\[ Y = \beta_0 + \sum_{i=1}^{p} \beta_i g_i(x) + \epsilon \]

where \( g_i(\cdot) \) are specified functions, not estimated.

Assume that \( \mathbb{E}(\epsilon) = 0, \text{Var}(\epsilon) = \sigma^2 \)

Often assume \( \epsilon \) is Gaussian, but this is not essential
Linear Regression

The Design Matrix:

\[
D = \begin{pmatrix}
1 & g_1(X_1) & g_2(X_1) & \cdots & g_p(X_1) \\
1 & g_1(X_2) & g_2(X_2) & \cdots & g_p(X_2) \\
\vdots & \vdots & \vdots & \ddots & \vdots \\
1 & g_1(X_n) & g_2(X_n) & \cdots & g_p(X_n)
\end{pmatrix}
\]

Then, \( L = D(D^TD)^{-1}D^T \) is the projection matrix.

Note that \( \nu = \text{trace}(L) = p + 1 \)
Linear Regression

Let $Y$ be a $n$-vector filled with $Y_1, Y_2, \ldots, Y_n$.

The Least Squares Estimates of $\beta$:

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

The Fitted Values:

$$\hat{Y} = LY = D\hat{\beta}$$

The Residuals:

$$\hat{\epsilon} = Y - \hat{Y} = (I - L)Y$$
Linear Regression

The estimates of $\beta$ given above minimize the sum of squared errors:

$$SSE = \sum_{i=1}^{n} \hat{\epsilon}_i^2 = \sum_{i=1}^{n} \left( Y_i - \hat{Y}_i \right)^2$$

If $\epsilon_i$ are Gaussian, then $\hat{\beta}$ is Gaussian, leading to confidence intervals, hypothesis tests
Linear regression in R is done via the command `lm()`

```r
> holdlm = lm(y ~ x)
```

Default is to include intercept term. If want to exclude, use

```r
> holdlm = lm(y ~ x - 1)
```
Linear Regression in R

Fitting the model \( Y = \beta_0 + \beta_1 x + \epsilon \)

The `summary()` shows the key output

```r
> holdlm = lm(y ~ x)
> summary(holdlm)

Coefficients:
          Estimate Std. Error t value Pr(>|t|)    
(Intercept)  -0.2468  0.1061  -2.326  0.0319 *  
   x            2.0671  0.1686  12.260 3.57e-10 ***
```

Note that \( \hat{\beta}_1 = 2.07 \), the SE for \( \hat{\beta}_1 \) is approximately 0.17.
### Linear Regression in R

Coefficients:

|            | Estimate | Std. Error | t value | Pr(>|t|) |
|------------|----------|------------|---------|----------|
| (Intercept)| -0.2468  | 0.1061     | -2.326  | 0.0319   * |
| x          | 2.0671   | 0.1686     | 12.260  | 3.57e-10 *** |

If the $\epsilon$ are Gaussian, then the last column gives the p-value for the test of $H_0: \beta_i = 0$ versus $H_1: \beta_i \neq 0$.

Also, if the $\epsilon$ are Gaussian,

$$\hat{\beta}_i \pm 2 \text{SE}(\hat{\beta}_i)$$

is an approximate 95% confidence interval for $\beta_i$. 
Always a good idea to look at plot of residuals versus fitted values. There should be no apparent pattern.

> plot(holdlm$residuals, holdlm$fitted.values)
Is it reasonable to assume $\epsilon$ are Gaussian? Check the normal probability plot of the residuals.

```r
> qqplot(holdlm$residuals)
> qqline(holdlm$residuals)
```
Simple to include more terms in the model:
> holdlm = lm(y ~ x + z)

Unfortunately, this does not work:
> holdlm = lm(y ~ x + x^2)

Instead, create a new variable and use that:
> x2 = x^2
> holdlm = lm(y ~ x + x2)
The Correlation Coefficient

A standard way of quantifying the strength of the linear relationship between two variables is via the (Pearson) correlation coefficient.

Sample Version:

\[ r = \frac{1}{n - 1} \sum_{i=1}^{n} X'_i \cdot Y'_i \]

where \( X'_i \) and \( Y'_i \) are the standardized versions of \( X_i \) and \( Y_i \), i.e.,

\[ X'_i = \frac{X_i - \overline{X}}{s_X}. \]
The Correlation Coefficient

Note that $-1 \leq r \leq 1$.

$r = 1$ indicates perfect \textit{linear}, increasing relationship
$r = -1$ indicates perfect \textit{linear}, decreasing relationship

Also, note that

$$\hat{\beta}_1 = r \left( \frac{s_Y}{s_X} \right)$$

In R: use \texttt{cor()}
The Correlation Coefficient

Examples of different scatter plots and values for \( r \) (Wikipedia)
A very popular, but overused, regression diagnostic is the coefficient of determination, denoted $R^2$

$$R^2 = 1 - \frac{\sum_{i=1}^{n} (Y_i - \hat{Y}_i)^2}{\sum_{i=1}^{n} (Y_i - \bar{Y})^2}$$

“The proportion of the variation explained by the model.”

Note that $0 \leq R^2 \leq 1$, and, $R^2 = r^2$.

But: Large $R^2$ does not imply that “the model is a good fit.”
“Anscombe’s Quartet” (1973)

Each of these has the same value for the marginal means and variances, same $r$, same $R^2$, same $\hat{\beta}_0$ and $\hat{\beta}_1$. 
Next. . .

Now we will begin our discussion of nonparametric regression.

Reconsider the SNe example from the start of the lecture.
Example: Type Ia Supernovae

Nonparametric estimate.
The Bias–Variance Tradeoff

Nonparametric Case:

Every nonparametric procedure requires a smoothing parameter $h$.

Consider the regression estimator based on local averaging:

$$\hat{f}(x) = \text{mean}\{Y_i : |X_i - x| \leq h\}.$$ 

Increasing $h$ yields smoother estimate $\hat{f}$.
What is “nonparametric?”

In the parametric case, the influence of assumptions is fixed.
In the nonparametric case, the influence of assumptions is controlled by $h$, where $h = O \left( \frac{1}{n^{1/5}} \right)$.
The Bias–Variance Tradeoff

Can formalize this via appropriate measure of error in estimator

Squared error loss:

\[ L(f(x), \hat{f}(x)) = (f(x) - \hat{f}(x))^2. \]

Mean squared error MSE (risk)

\[ \text{MSE} = R(f(x), \hat{f}(x)) = \mathbb{E}(L(f(x), \hat{f}(x))). \]
The Bias–Variance Tradeoff

The key result:

\[ R(f(x), \hat{f}(x)) = \text{bias}_x^2 + \text{variance}_x \]

where

\[ \text{bias}_x = \mathbb{E}(\hat{f}(x)) - f(x) \]
\[ \text{variance}_x = \text{Var}(\hat{f}(x)) \]

Often written as

\[ \text{MSE} = \text{BIAS}^2 + \text{VARIANCE}. \]
The Bias–Variance Tradeoff

Mean Integrated Squared Error:

\[ MISE = \int R(f(x), \hat{f}(x)) \, dx \]

\[ = \int \text{bias}_x^2 \, dx + \int \text{variance}_x \, dx \]

Empirical version:

\[ \frac{1}{n} \sum_{i=1}^{n} R(f(x_i), \hat{f}(x_i)). \]
The Bias–Variance Tradeoff

- Bias squared
- Variance
- Risk

Introduction to Regression – p. 36/100
The Bias–Variance Tradeoff

For nonparametric procedures, when $x$ is one-dimensional,

$$MISE \approx c_1 h^4 + \frac{c_2}{nh}$$

which is minimized at

$$h = O \left( \frac{1}{n^{1/5}} \right)$$

Hence,

$$MISE = O \left( \frac{1}{n^{4/5}} \right)$$

whereas, for parametric problems, when the model is correct,

$$MISE = O \left( \frac{1}{n} \right)$$
Now we will begin our discussion of nonparametric regression procedures.

But, these nonparametric procedures share a common structure with linear regression.

Namely, they are all linear smoothers.
Linear Smoothers

For a linear smoother:

\[ \hat{f}(x) = \sum_{i} Y_i \ell_i(x) \]

for some weights:

\[ \ell_1(x), \ldots, \ell_n(x) \]

The vector of weights depends on the target point \( x \).
Linear Smoothers

If \( \hat{f}(x) = \sum_{i=1}^{n} \ell_i(x)Y_i \) then

\[
\begin{pmatrix}
\hat{Y}_1 \\
\vdots \\
\hat{Y}_n \\
\hat{\hat{Y}}
\end{pmatrix} = \begin{pmatrix}
\hat{f}(x_1) \\
\vdots \\
\hat{f}(x_n)
\end{pmatrix} = \begin{pmatrix}
\ell_1(x_1) & \ell_2(x_1) & \cdots & \ell_n(x_1) \\
\ell_1(x_2) & \ell_2(x_2) & \cdots & \ell_n(x_2) \\
\vdots & \vdots & \ddots & \vdots \\
\ell_1(x_n) & \ell_2(x_n) & \cdots & \ell_n(x_n)
\end{pmatrix} \begin{pmatrix}
Y_1 \\
\vdots \\
Y_n
\end{pmatrix}
\]

\[ \hat{Y} = LY \]

The effective degrees of freedom is:

\[ \nu = \text{trace}(L) = \sum_{i=1}^{n} L_{ii}. \]

Can be interpreted as the number of parameters in the model. Recall that \( \nu = p + 1 \) for linear regression.
Linear Smoothers

Consider the Extreme Cases:

When $\ell_i(X_j) = 1/n$ for all $i, j$,

$$\hat{f}(X_j) = \bar{Y} \quad \text{for all } j$$

and $\nu = 1$.

When $\ell_i(X_j) = \delta_{ij}$ for all $i, j$,

$$\hat{f}(X_j) = Y_j$$

and $\nu = n$. 
To begin the discussion of nonparametric procedures, we start with a simple, but naive, approach: binning.
Binning

Divide the $x$-axis into bins $B_1, B_2, \ldots$ of width $h$. $\hat{f}$ is a step function based on averaging the $Y_i$'s in each bin:

$$\text{for } x \in B_j : \quad \hat{f}(x) = \text{mean}\left\{ Y_i : X_i \in B_j \right\}.$$ 

The (arbitrary) choice of the boundaries of the bins can affect inference, especially when $h$ large.
Binning

WMAP data, binned estimate with \( h = 15 \).
Binning

WMAP data, binned estimate with $h = 50$. 

![Graph showing binned data with $h = 50$.]
Binning

WMAP data, binned estimate with $h = 75$. 
A step beyond binning is to take local averages to estimate the regression function.
Local Averaging

For each $x$ let

$$N_x = \left[ x - \frac{h}{2}, x + \frac{h}{2} \right].$$

This is a moving window of length $h$, centered at $x$. Define

$$\hat{f}(x) = \text{mean}\left\{ Y_i : X_i \in N_x \right\}.$$

This is like binning but removes the arbitrary boundaries.
Local Averaging

WMAP data, local average estimate with $h = 15$. 

![Graph showing data analysis with local averaging and comparison to binned data.]
Local Averaging

WMAP data, local average estimate with $h = 50$. 
Local Averaging

WMAP data, local average estimate with $h = 75$. 

-4000  -2000  0  2000  4000  6000  8000

Binned

Local Averaging
Local averaging is a special case of kernel regression.
The local average estimator can be written:

\[ \hat{f}(x) = \frac{\sum_{i=1}^{n} Y_i \ K \left( \frac{x-X_i}{h} \right)}{\sum_{i=1}^{n} K \left( \frac{x-X_i}{h} \right)} \]

where

\[ K(x) = \begin{cases} 1 & |x| < 1/2 \\ 0 & \text{otherwise.} \end{cases} \]

Can improve this by using a function \( K \) which is smoother.
Kernels

A kernel is any smooth function $K$ such that $K(x) \geq 0$ and

$$\int K(x) \, dx = 1, \quad \int xK(x) \, dx = 0 \quad \text{and} \quad \sigma_K^2 \equiv \int x^2 K(x) \, dx > 0.$$ 

Some commonly used kernels are the following:

- the boxcar kernel: $K(x) = \frac{1}{2}I(|x| < 1)$,

- the Gaussian kernel: $K(x) = \frac{1}{\sqrt{2\pi}} e^{-x^2/2}$,

- the Epanechnikov kernel: $K(x) = \frac{3}{4}(1 - x^2)I(|x| < 1)$,

- the tricube kernel: $K(x) = \frac{70}{81}(1 - |x|^3)^3 I(|x| < 1)$. 

Kernels
Kernel Regression

\[ \hat{f}(x) = \frac{\sum_{i=1}^{n} Y_i \ K \left( \frac{x-X_i}{h} \right)}{\sum_{i=1}^{n} K \left( \frac{x-X_i}{h} \right)} \]

We can write this as

\[ \hat{f}(x) = \sum_i Y_i \ \ell_i(x) \]

where

\[ \ell_i(x) = \frac{K \left( \frac{x-X_i}{h} \right)}{\sum_{i=1}^{n} K \left( \frac{x-X_i}{h} \right)}. \]
Kernel Regression

WMAP data, kernel regression estimates, $h = 15$. 

![Graph showing different kernel regression estimates with a legend indicating the types of kernels used: boxcar kernel, Gaussian kernel, tricube kernel.](image-url)
Kernel Regression

WMAP data, kernel regression estimates, $h = 50$. 

![Graph showing WMAP data with kernel regression estimates for different kernels: boxcar, Gaussian, and tricube. The graph plots $C_l$ against $l$.](image-url)
Kernel Regression

WMAP data, kernel regression estimates, \( h = 75 \).
Kernel Regression

\[
\text{MISE} \approx \frac{h^4}{4} \left( \int x^2 K(x) \, dx \right)^2 \int \left( f''(x) + 2f'(x) \frac{g'(x)}{g(x)} \right)^2 \, dx \\
+ \frac{\sigma^2}{nh} \int K^2(x) \, dx \int \frac{1}{g(x)} \, dx.
\]

where \( g(x) \) is the density for \( X \).

What is especially notable is the presence of the term

\[
2f'(x) \frac{g'(x)}{g(x)} = \text{design bias}.
\]

Also, bias is large near the boundary. We can reduce these biases using local polynomials.
Kernel Regression

WMAP data, \( h = 50 \). Note the boundary bias.
The deficiencies of kernel regression can be addressed by considering models which, on small scales, model the regression function as a low order polynomial.
Local Polynomial Regression

Recall polynomial regression:

\[ \hat{f}(x) = \hat{\beta}_0 + \hat{\beta}_1 x + \hat{\beta}_2 x^2 + \cdots + \hat{\beta}_p x^p \]

where \( \hat{\beta} = (\hat{\beta}_0, \ldots, \hat{\beta}_p) \) are obtained by least squares:

\[
\text{minimize} \quad \sum_{i=1}^{n} \left( Y_i - [\beta_0 + \beta_1 x + \beta_2 x^2 + \cdots + \beta_p x^p] \right)^2
\]
Local Polynomial Regression

Local polynomial regression: approximate $f(x)$ locally by a different polynomial for every $x$:

$$f(u) \approx \beta_0(x) + \beta_1(x)(u - x) + \beta_2(x)(u - x)^2 + \cdots + \beta_p(x)(u - x)^p$$

for $u$ near $x$. Estimate $(\hat{\beta}_0(x), \ldots, \hat{\beta}_p(x))$ by local least squares: minimize

$$\sum_{i=1}^{n} (Y_i - [\beta_0(x) + \beta_1(x)x + \beta_2(x)x^2 + \cdots + \beta_p(x)x^p])^2 K \left( \frac{x - X_i}{h} \right)$$

kernel

$$\hat{f}(x) = \hat{\beta}_0(x)$$
Local Polynomial Regression

Taking $p = 0$ yields the kernel regression estimator:

$$\hat{f}_n(x) = \sum_{i=1}^{n} \ell_i(x)Y_i$$

$$\ell_i(x) = \frac{K\left(\frac{x-x_i}{h}\right)}{\sum_{j=1}^{n} K\left(\frac{x-x_j}{h}\right)}.$$

Taking $p = 1$ yields the local linear estimator. This is the best, all-purpose smoother.

Choice of Kernel $K$: not important
Choice of bandwidth $h$: crucial
Local Polynomial Regression

The local polynomial regression estimate is

\[ \hat{f}_n(x) = \sum_{i=1}^{n} \ell_i(x)Y_i \]

where \( \ell(x)^T = (\ell_1(x), \ldots, \ell_n(x)) \),

\[ \ell(x)^T = e_1^T (X_x^T W_x X_x)^{-1} X_x^T W_x, \]

\( e_1 = (1, 0, \ldots, 0)^T \) and \( X_x \) and \( W_x \) are defined by

\[
X_x = \begin{pmatrix}
1 & x_1 - x & \cdots & \frac{(x_1 - x)^p}{p!} \\
1 & x_2 - x & \cdots & \frac{(x_2 - x)^p}{p!} \\
\vdots & \vdots & \ddots & \vdots \\
1 & x_n - x & \cdots & \frac{(x_n - x)^p}{p!}
\end{pmatrix} \quad \text{and} \quad W_x = \begin{pmatrix}
K \left( \frac{x - X_1}{h} \right) & 0 & \cdots & 0 \\
\vdots & \vdots & \ddots & \vdots \\
0 & 0 & 0 & K \left( \frac{x - X_n}{h} \right)
\end{pmatrix},
\]
Local Polynomial Regression

Note that \( \hat{f}(x) = \sum_{i=1}^{n} \ell_i(x) Y_i \) is a linear smoother. Define \( \hat{Y} = (\hat{Y}_1, \ldots, \hat{Y}_n) \) where \( \hat{Y}_i = \hat{f}(X_i) \). Then

\[ \hat{Y} = LY \]

where \( L \) is the smoothing matrix:

\[
L = \begin{pmatrix}
\ell_1(X_1) & \ell_2(X_1) & \cdots & \ell_n(X_1) \\
\ell_1(X_2) & \ell_2(X_2) & \cdots & \ell_n(X_2) \\
\vdots & \vdots & \ddots & \vdots \\
\ell_1(X_n) & \ell_2(X_n) & \cdots & \ell_n(X_n)
\end{pmatrix}.
\]

The effective degrees of freedom is:

\[ \nu = \text{trace}(L) = \sum_{i=1}^{n} L_{ii}. \]
Theoretical Aside

Why local linear \((p = 1)\) is better than kernel \((p = 0)\). Both have (approximate) variance

\[
\frac{\sigma^2(x)}{g(x)nh} \int K^2(u)du
\]

The kernel estimator has bias

\[
h^2 \left( \frac{1}{2} f''(x) + \frac{f'(x)g'(x)}{g(x)} \right) \int u^2 K(u)du
\]

whereas the local linear estimator has asymptotic bias

\[
h^2 \frac{1}{2} f''(x) \int u^2 K(u)du
\]

The local linear estimator is free from design bias. At the boundary points, the kernel estimator has asymptotic bias of \(O(h)\) while the local linear estimator has bias \(O(h^2)\).
Parametric procedures force the choice $p$, the number of predictors.

Nonparametric procedures force the choice of the smoothing parameter $h$.

Next we will discuss how this can be selected in a data-driven manner via cross-validation.
Choosing \( p \) and \( h \)

Estimate the risk

\[
\frac{1}{n} \sum_{i=1}^{n} \mathbb{E}(\hat{f}(X_i) - f(X_i))^2
\]

with the \textbf{leave-one-out cross-validation score}:

\[
CV = \frac{1}{n} \sum_{i=1}^{n} (Y_i - \hat{f}(-i)(X_i))^2
\]

where \( \hat{f}(-i) \) is the estimator obtained by omitting the \( i^{\text{th}} \) pair \((X_i, Y_i)\).
Choosing $p$ and $h$

Amazing shortcut formula:

$$CV = \frac{1}{n} \sum_{i=1}^{n} \left( \frac{Y_i - \hat{f}_n(x_i)}{1 - L_{ii}} \right)^2.$$ 

An commonly used approximation is GCV (generalized cross-validation):

$$GCV = \frac{1}{n \left(1 - \frac{\nu}{n}\right)^2} \sum_{i=1}^{n} (Y_i - \hat{f}(X_i))^2$$

$$\nu = \text{trace}(L).$$
Next, we will go over a bit of how to perform local polynomial regression in R.
Using `locfit()`

Need to include the `locfit` library:
> install.packages("locfit")
> library(locfit)
> result = locfit(y~x, alpha=c(0, 1.5), deg=1)

`y` and `x` are vectors

the second argument to `alpha` gives the bandwidth \( (h) \)

the first argument to `alpha` specifies the nearest neighbor fraction, an alternative to the bandwidth

`fitted(result)` gives the fitted values, \( \hat{f}(X_i) \)

`residuals(result)` gives the residuals, \( Y_i - \hat{f}(X_i) \)
Using `locfit()`

degree=1, n=1000, sigma=0.1

\[ y(x) = \sqrt{x(1-x)} \sin((2.1\pi)/(x+0.2)) \]

- red: estimate using \( h = 0.005 \)
- green: estimate using \( h = 0.015 \) (gcv optimal)

\[ x \in [0,1] \]
Using `locfit()`

degree=1, n=1000, sigma=0.1

\[
y = \sqrt{x(1-x)} \sin((2.1\pi)/(x + 0.2))
\]

- Red: estimate using \( h = 0.1 \)
- Green: estimate using \( h = 0.1 \)
- Blue: estimate using gcv optimal \( h = 0.015 \)
Using `locfit()`

degree=1, n=1000, sigma=0.5

![Graph showing results of using `locfit()` with degree=1, n=1000, sigma=0.5. The graph plots a function \((x + 5\exp(-4x^2))\) and estimates using different bandwidths: h = 0.05 and gcv optimal h = 0.174. The graph compares the original function with the estimated curves, highlighting the effect of different bandwidths on the fit.]
Using `locfit()`

degree=1, n=1000, sigma=0.5

\[(x + 5\exp(-4x^2))\]

- estimate using \(h = 0.5\)
- estimate using gcv optimal \(h = 0.174\)
Example

WMAP data, local linear fit, $h = 15$.  

![Graph showing WMAP data with local linear fit using $h = 15$ and gcv optimal $h = 63.012$.]
Example

WMAP data, local linear fit, \( h = 125 \).
We should quantify the amount of error in our estimator for the regression function. Next we will look at how this can be done using local polynomial regression.
Variance Estimation

Let

\[ \hat{\sigma}^2 = \frac{\sum_{i=1}^{n} (Y_i - \hat{f}(x_i))^2}{n - 2\nu + \tilde{\nu}} \]

where

\[ \nu = \text{tr}(L), \quad \tilde{\nu} = \text{tr}(L^T L) = \sum_{i=1}^{n} ||\ell(x_i)||^2. \]

If \( f \) is sufficiently smooth, then \( \hat{\sigma}^2 \) is a consistent estimator of \( \sigma^2 \).
Variance Estimation

For the WMAP data, using local linear fit.

> wmap = read.table("wmap.dat", header=T)
> opth = 63.0

> locfitwmap = locfit(wmap$Cl[1:700]~wmap$ell[1:700],
  alpha=c(0,opth), deg=1)
> nu = as.numeric(locfitwmap$dp[6])
> nutilde = as.numeric(locfitwmap$dp[7])

> sigmasqrhat = sum(residuals(locfitwmap)^2)/(700-2*nu+nutilde)
> sigmasqrhat
[1] 1122214

But, does not seem reasonable to assume homoscedasticity...
Allow $\sigma$ to be a function of $x$:

$$Y_i = f(x_i) + \sigma(x_i)\epsilon_i.$$ 

Let $Z_i = \log(Y_i - f(x_i))^2$ and $\delta_i = \log \epsilon_i^2$. Then,

$$Z_i = \log(\sigma^2(x_i)) + \delta_i.$$ 

This suggests estimating $\log \sigma^2(x)$ by regressing the log squared residuals on $x$. 
Variance Estimation

1. Estimate $f(x)$ with any nonparametric method to get an estimate $\hat{f}_n(x)$.

2. Define $Z_i = \log(Y_i - \hat{f}_n(x_i))^2$.

3. Regress the $Z_i$'s on the $x_i$'s (again using any nonparametric method) to get an estimate $\hat{q}(x)$ of $\log \sigma^2(x)$ and let

$$\hat{\sigma}^2(x) = e^{\hat{q}(x)}.$$
Example

WMAP data, log squared residuals, local linear fit, $h = 63.0$. 
Example

With local linear fit, $h = 130$, chosen via GCV
Example

Estimating $\sigma(x)$:

![Graph showing the estimation of $\sigma(x)$]
Confidence Bands

Recall that

\[ \hat{f}(x) = \sum_i Y_i \ell_i(x) \]

so

\[ \text{Var}(\hat{f}(x)) = \sum_i \sigma^2(X_i) \ell_i^2(x). \]

An approximate $1 - \alpha$ confidence interval for $f(x)$ is

\[ \hat{f}(x) \pm z_{\alpha/2} \sqrt{\sum_i \hat{\sigma}^2(X_i) \ell_i^2(x)}. \]

When $\sigma(x)$ is smooth, we can approximate

\[ \sqrt{\sum_i \hat{\sigma}^2(X_i) \ell_i^2(x)} \approx \hat{\sigma}(x) \| \ell(x) \|. \]
Confidence Bands

Two caveats:

1. \( \hat{f} \) is biased so this is really an interval for \( \mathbb{E}(\hat{f}(x)) \). Result: bands can miss sharp peaks in the function.

2. Pointwise coverage does not imply simultaneous coverage for all \( x \).

Solution for 2 is to replace \( z_{\alpha/2} \) with a larger number (Sun and Loader 1994). \texttt{locfit()} does this for you.
> diaghat = predict.locfit(locfitwmap, where="data", what="infl")
> normell = predict.locfit(locfitwmap, where="data", what="vari")

diaghat will be $L_{ii}, i = 1, 2, \ldots, n.$
normell will be $||\ell(X_i)||, i = 1, 2, \ldots, n$

The Sun and Loader replacement for $z_{\alpha/2}$ is found using
kappa0(locfitwmap)$crit.val.$
Example

95% pointwise confidence bands:

- **Cl**
  - **estimate**
  - **red**: 95% pointwise band assuming constant variance
  - **blue**: 95% pointwise band using nonconstant variance

![Graph showing 95% pointwise confidence bands with labels for the estimate, band assuming constant variance, and band using nonconstant variance.](image-url)
95% simultaneous confidence bands:

- **Red line**: 95% simultaneous band assuming constant variance
- **Blue line**: 95% simultaneous band using nonconstant variance

**Legend**:
- **Black line**: Estimate

The graph shows the comparison between the confidence bands under constant and nonconstant variance assumptions. The x-axis represents the index, and the y-axis represents the confidence interval (CI) values.
We discuss the use of regression splines. These are a well-motivated choice of basis for constructing regression functions.
Idea: expand $f$ as

$$f(x) = \sum_j \beta_j \psi_j(x)$$

where $\psi_1(x), \psi_2(x), \ldots$ are specially chosen, known functions. Then estimate $\beta_j$ and set

$$f(x) = \sum_j \hat{\beta}_j \psi_j(x).$$

Here we consider a popular version, splines
Splines and Penalization

Define $\hat{f}_n$ to be the function that minimizes

$$M(\lambda) = \sum_i (Y_i - \hat{f}_n(x_i))^2 + \lambda \int (f''(x))^2 dx.$$ 

- $\lambda = 0 \implies \hat{f}_n(X_i) = Y_i$ (no smoothing)
- $\lambda = \infty \implies \hat{f}_n(x) = \hat{\beta}_0 + \hat{\beta}_1 x$ (linear)
- $0 < \lambda < \infty \implies \hat{f}_n(x) =$ cubic spline with knots at $X_i$.

A cubic spline is a continuous function $f$ such that

1. $f$ is a cubic polynomial between the $X_i$’s
2. $f$ has continuous first and second derivatives at the $X_i$’s.
Basis For Splines

Define $(z)_+ = \max\{z, 0\}$, $N = n + 4$,

$$
\psi_1(x) = 1 \quad \psi_2(x) = x \quad \psi_3(x) = x^2 \quad \psi_4(x) = x^3 \quad \\
\psi_5(x) = (x - X_1)_+^3 \quad \psi_6(x) = (x - X_2)_+^3 \quad \cdots \quad \psi_N(x) = (x - X_n)_+^3.
$$

These functions form a basis for the splines: we can write

$$
f(x) = \sum_{j=1}^{N} \beta_j \psi_j(x).
$$

(For numerical calculations it is actually more efficient to use other spline bases.) We can thus write

$$
(1) \quad \hat{f}_n(x) = \sum_{j=1}^{N} \hat{\beta}_j \psi_j(x),
$$
Basis For Splines

We can now rewrite the minimization as follows:

\[
\text{minimize : } (Y - \Psi \beta)^T (Y - \Psi \beta) + \lambda \beta^T \Omega \beta
\]

where \( \Psi_{ij} = \psi_j(X_i) \) and \( \Omega_{jk} = \int \psi''_j(x) \psi''_k(x) \, dx \). The value of \( \beta \) that minimizes this is

\[
\hat{\beta} = (\Psi^T \Psi + \lambda \Omega)^{-1} \Psi^T Y.
\]

The smoothing spline \( \hat{f}_n(x) \) is a linear smoother, that is, there exist weights \( \ell(x) \) such that

\[
\hat{f}_n(x) = \sum_{i=1}^n Y_i \ell_i(x).
\]
Basis functions with 5 knots.
Basis For Splines

Same span as previous slide, the B-spline basis, 5 knots:
> smosplresult = smooth.spline(x,y, cv=FALSE, all.knots=TRUE)

If cv=TRUE, then cross-validation used to choose $\lambda$; if cv=FALSE, gcv is used

If all.knots=TRUE, then knots are placed at all data points; if all.knots=FALSE, then set nknots to specify the number of knots to be used. Using fewer knots eases the computational cost.

predict(smosplresult) gives fitted values.
Example

WMAP data, $\lambda$ chosen using GCV.
Finally, we address the challenges of fitting regression models when the predictor $x$ is of large dimension.
Recall: The Bias–Variance Tradeoff

For nonparametric procedures, when $x$ is one-dimensional,

$$MISE \approx c_1 h^4 + \frac{c_2}{nh}$$

which is minimized at

$$h = O \left( \frac{1}{n^{1/5}} \right)$$

Hence,

$$MISE = O \left( \frac{1}{n^{4/5}} \right)$$

whereas, for parametric problems, when the model is correct,

$$MISE = O \left( \frac{1}{n} \right)$$
The Curse of Dimensionality

For nonparametric procedures, when $x$ is $d$-dimensional,

$$\text{MISE} \approx c_1 h^d + \frac{c_2}{nh^d}$$

which is minimized at

$$h = O \left( \frac{1}{n^{1/(4+d)}} \right)$$

Hence,

$$\text{MISE} = O \left( \frac{1}{n^{4/(4+d)}} \right)$$

whereas, for parametric problems, when the model is correct, still,

$$\text{MISE} = O \left( \frac{1}{n} \right)$$
Multiple Regression

\[ Y = f(X_1, X_2, \ldots, X_d) + \epsilon \]

The curse of dimensionality:
Optimal rate of convergence for \( d = 1 \) is \( n^{-4/5} \). In \( d \) dimensions the optimal rate of convergence is \( n^{-4/(4+d)} \). Thus, the sample size \( m \) required for a \( d \)-dimensional problem to have the same accuracy as a sample size \( n \) in a one-dimensional problem is \( m \propto n^{cd} \) where
\[
c = (4 + d)/(5d) > 0.
\]

To maintain a given degree of accuracy of an estimator, the sample size must increase exponentially with the dimension \( d \).

Put another way, confidence bands get very large as the dimension \( d \) increases.
Multiple Local Linear

Given a nonsingular positive definite $d \times d$ bandwidth matrix $H$, we define

$$K_H(x) = \frac{1}{|H|^{1/2}} K(H^{-1/2}x).$$

Often, one scales each covariate to have the same mean and variance and then we use the kernel

$$h^{-d} K(||x||/h)$$

where $K$ is any one-dimensional kernel. Then there is a single bandwidth parameter $h$. 
Multiple Local Linear

At a target value \( x = (x_1, \ldots, x_d)^T \), the local sum of squares is given by

\[
\sum_{i=1}^{n} w_i(x) \left( Y_i - a_0 - \sum_{j=1}^{d} a_j (x_{ij} - x_j) \right)^2
\]

where

\[
w_i(x) = K(||x_i - x||/h).
\]

The estimator is \( \hat{f}_n(x) = \hat{a}_0 \hat{a} = (\hat{a}_0, \ldots, \hat{a}_d)^T \) is the value of \( a = (a_0, \ldots, a_d)^T \) that minimizes the weighted sums of squares.
Multiple Local Linear

The solution \( \hat{a} \) is

\[
\hat{a} = (X_x^T W_x X_x)^{-1} X_x^T W_x Y
\]

where

\[
X_x = \begin{pmatrix}
1 & (x_{11} - x_1) & \cdots & (x_{1d} - x_d) \\
1 & (x_{21} - x_1) & \cdots & (x_{2d} - x_d) \\
\vdots & \vdots & \ddots & \vdots \\
1 & (x_{n1} - x_1) & \cdots & (x_{nd} - x_d)
\end{pmatrix}
\]

and \( W_x \) is the diagonal matrix whose \((i, i)\) element is \( w_i(x) \).
A Compromise: Additive Models

\[ Y = \alpha + \sum_{j=1}^{d} f_j(x_j) + \epsilon \]

Usually take \( \hat{\alpha} = \overline{Y} \). Then estimate the \( f_j \)'s by backfitting.

1. set \( \hat{\alpha} = \overline{Y} \), \( \hat{f}_1 = \cdots \hat{f}_d = 0 \).

2. Iterate until convergence: for \( j = 1, \ldots, d \):
   - Compute \( \tilde{Y}_i = Y_i - \hat{\alpha} - \sum_{k \neq j} \hat{f}_k(X_i), i = 1, \ldots, n \).
   - Apply a smoother to \( \tilde{Y}_i \) on \( X_j \) to obtain \( \hat{f}_j \).
   - Set \( \hat{f}_j(x) \) equal to \( \hat{f}_j(x) - n^{-1} \sum_{i=1}^{n} \hat{f}_j(x_i) \).

The last step ensures that \( \sum_i \hat{f}_j(X_i) = 0 \) (identifiability).
To Sum Up...

Classic linear regression is computationally and theoretically simple

Nonparametric procedures for regression offer increased flexibility

Careful consideration of MISE suggests use of local polynomial fits

Smoothing parameters can be chosen objectively

High-dimensional data present a challenge