Local regression

Patrick Breheny

November 4

Introduction

- For the remainder of the course, we will focus on nonparametric regression and classification
- The regression problem involves modeling how the expected value of a response y changes in response to changes in an explanatory variable x:

$$\mathbb{E}(y|x) = f(x)$$

- Linear regression, as its name implies, assumes a linear relationship; namely, that $f(x) = \beta_0 + \beta_1 x$

Parametric vs. nonparametric approaches

- This reduction of a complicated function to a simple form with a small number of unknown parameters is very similar to the parametric approach to estimation and inference involving the unknown distribution function
- The nonparametric approach, in contrast, is to make as few assumptions about the regression function f as possible
- Instead, we will try to use the data as much as possible to learn about the potential shape of f – allowing f to be very flexible, yet smooth

Simple local models

- One way to achieve this flexibility is by fitting a different, simple model separately at every point x_0 in much the same way that we used kernels to estimate density
- Once again, this is done using only those observations close to x_0 to fit the simple model, with a continuous kernel to enforce that f is smooth as a function of x_0

The Nadaraya-Watson kernel estimator

- The simplest local model is the local average:

$$\hat{f}(x_0) = \frac{\sum_i y_i I(|x_i - x_0| < h)}{\sum_i I(|x_i - x_0| < h)}$$

- However, as we saw with kernel density estimates, this leads to a discontinuous estimate
- We can therefore generalize the above to the following, known as the *Nadaraya-Watson kernel estimator*:

$$\hat{f}(x_0) = \frac{\sum_i y_i K_h(x_i, x_0)}{\sum_i K_h(x_i, x_0)},$$

where $K_h(x_i, x_0) = K(\frac{x_i - x_0}{h})$ is the kernel, and if K is continuous, then so is $\hat{f}$

Expected loss for regression

- As with kernel density estimates, we need to estimate the bandwidth h, which controls the degree of smoothing
- Expected loss is defined slightly differently for regression than density estimation
- Because it is customary to treat x as fixed in regression, instead of integrating over x to obtain the expected loss, we average over the observed values of x:

$$\mathbb{E}L(f, \hat{f}) = \frac{1}{n} \sum_i \mathbb{E}L(f(x_i), \hat{f}(x_i))$$

Expected prediction error

- The average expected loss is particularly convenient in regression
- Letting $(y_i - \hat{f}(x_i))^2$ be the *squared prediction error*, and calling

$$EPE = \mathbb{E}\left\{\frac{1}{n}\sum_i (y_i - \hat{f}(x_i))^2\right\}$$

the *expected prediction error*, we have

$$EPE = \mathbb{E}L(f, \hat{f}) + \sigma^2,$$

where σ^2 is the variance of y

- Thus, the expected prediction error and the expected loss are equal up to a constant

Cross-validation

- This is attractive because prediction error is easy to evaluate via cross-validation
- Specifically, we can estimate the expected prediction error with

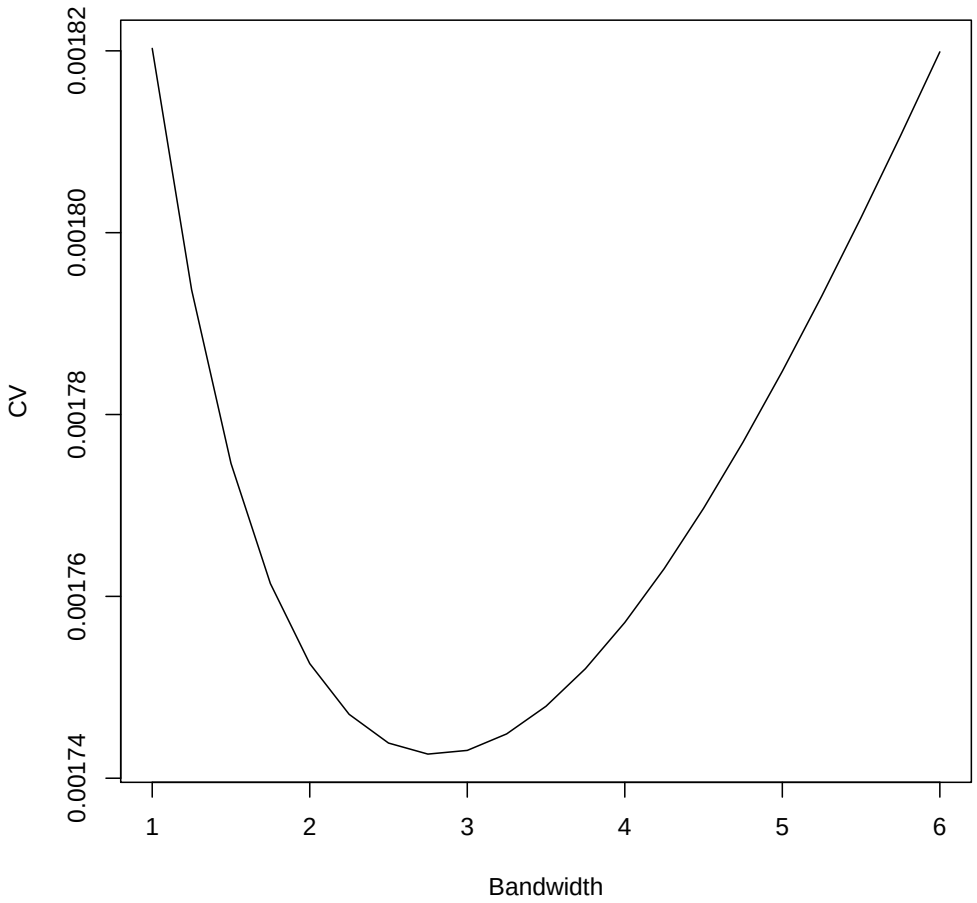
$$CV = \frac{1}{n} \sum_i (y_i - \hat{f}_{(-i)}(x_i))^2,$$

  where $\hat{f}_{(-i)}$ is the estimate of f obtained by omitting the pair $\{x_i, y_i\}$
- Furthermore, as we will see, one can obtain a closed form expression for the leave-one-out cross validation score above for any "linear smoother", without actually refitting the model n times

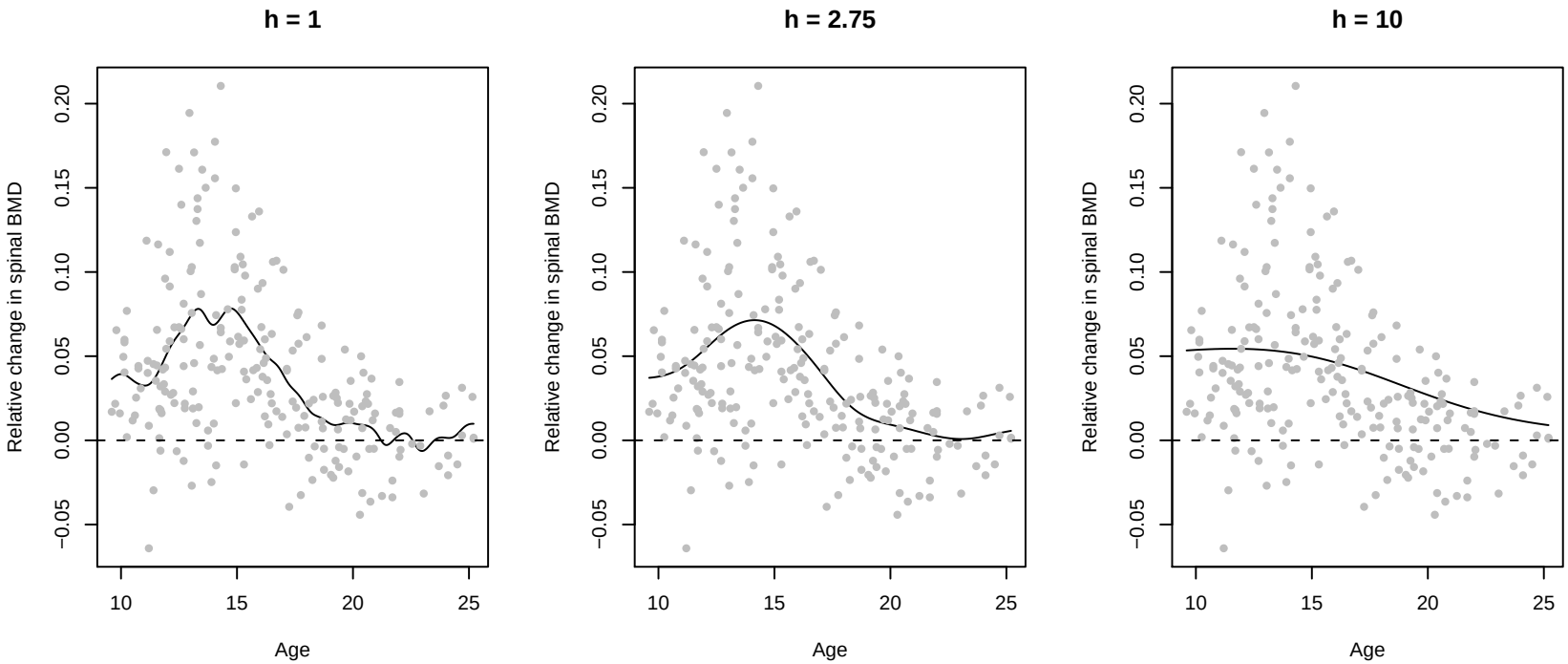
Bone mineral density data

- As an example of a real data set with an interesting change in $\mathbb{E}(y|x)$ as a function of x, we will look at a study of changes in bone mineral density in adolescents
- The outcome is the difference in spinal bone mineral density, taken on two consecutive visits, divided by the average of the two measurements
- Age is the average age over the two visits
- A person's bone mineral density generally increases until the individual is done growing, then remains relatively constant until old age

Cross-validation to choose bandwidth

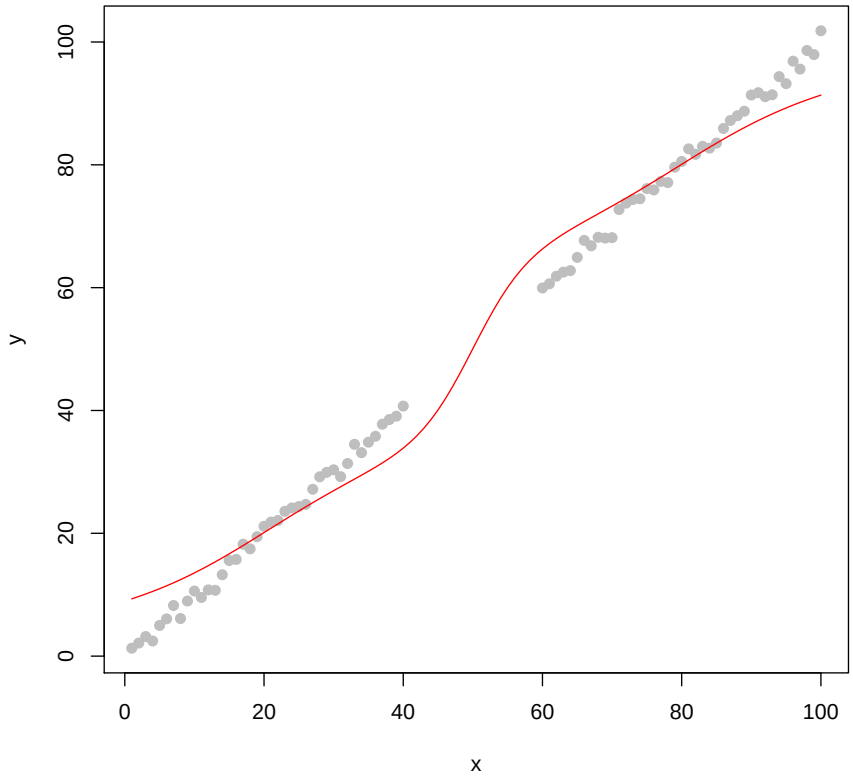


Estimates of the regression function



The problem with kernel weighted averages

Unfortunately, the Nadaraya-Watson kernel estimator suffers from bias, both at the boundaries and in the interior when the x_i's are not uniformly distributed:



Loess

- This arises due to the asymmetry effect of the kernel in these regions
- However, we can (up to first order) eliminate this problem by fitting straight lines locally, instead of constants
- In locally weighted regression, also known as *lowess* or *loess*, we solve a separate weighted least squares problem at each target point x_0:

$$(\hat{\alpha}, \hat{\beta}) = \text{arg min}_{\alpha,\beta} \sum_i K_h(x_0, x_i)(y_i - \alpha - x_i\beta)^2$$

- The estimate is then $\hat{f}(x_0) = \hat{\alpha} + x_0\hat{\beta}$

Loess is a linear smoother

- Let $\mathbf{X}$ denote the $n \times 2$ matrix with ith row $(1, x_i - x_0)$, and $\mathbf{W}$ denote the $n \times n$ diagonal matrix with ith diagonal element $w_i(x_0) = K_h(x_0, x_i)$
- Then,

$$\begin{aligned}\hat{f}(x_0) &= e_1'[\mathbf{X}'\mathbf{W}\mathbf{X}]^{-1}\mathbf{X}'\mathbf{W}\mathbf{y} \\ &= \sum_i l_i(x_0) y_i,\end{aligned}$$

where $e_1 = (1, 0)' = (1, x_0 - x_0)'$ and it is important to keep in mind that both $\mathbf{X}$ and $\mathbf{W}$ depend implicitly on x_0

Loess is a linear smoother (cont'd)

- Furthermore,

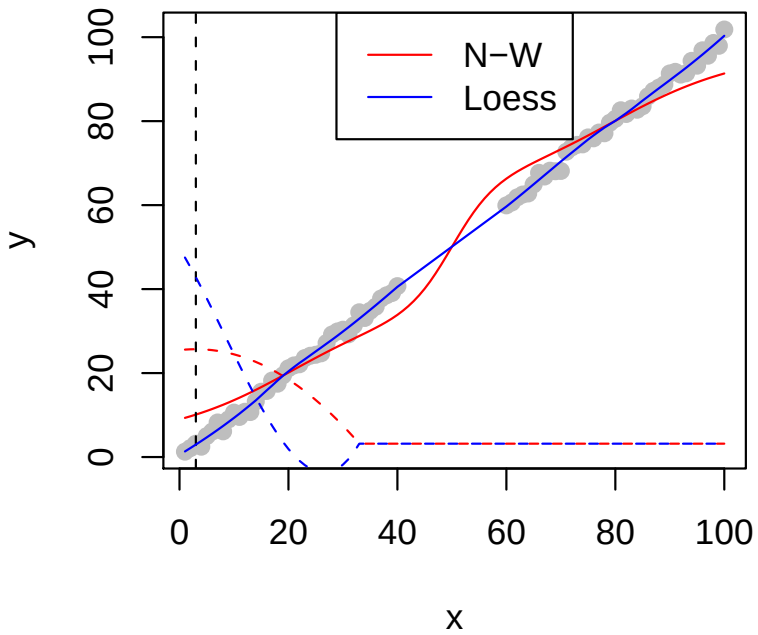
$$l_i(x_0) = \frac{w_i S_2 - w_i(x_i - x_0)S_1}{\sum_j \{w_j S_2 - w_j(x_j - x_0)S_1\}},$$

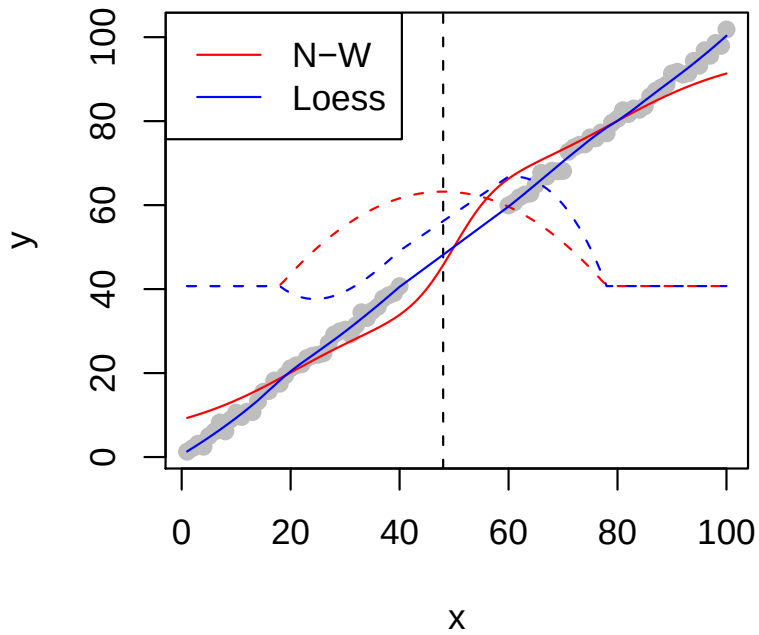
  where $S_k = \sum_i w_i(x_i - x_0)^k$
- Thus, $\sum_i l_i(x_0) = 1$ for all x_0
- Finally, note that $l_i(x_0)$ does not depend on $\{y_i\}$
- Thus, our estimate is a weighted linear combination of the y_i's; such estimates of f are called *linear smoothers*

Effective kernels

- Furthermore, the linear smoother in local linear regression is performing weighted local averaging with the weights, determined by $\{l_i(x_0)\}$, forming an *effective kernel* (also called the *equivalent kernel*)
- Before the development of loess, a fair amount of research focused on deriving adaptive modifications to kernels in order to alleviate the bias that we previously discussed
- However, local linear regression automatically modifies the kernel in such a way that this bias is largely eliminated, a phenomenon known as *automatic kernel carpentry*

Automatic kernel carpentry





The smoothing matrix

- Recall that loess is a linear smoother; thus,

$$\hat{\mathbf{y}} = \mathbf{L}\mathbf{y},$$

  where $\mathbf{L}$ is called the *smoothing matrix* whose elements consists of the linear weights $l_j(x_i)$
- Having our predictions take on this linear form greatly simplifies cross-validation
- Without refitting anything, we can obtain $\hat{f}_{(-i)}(x_i)$ via

$$\hat{f}_{(-i)}(x_i) = \sum_{j \neq i} \frac{l_{ij} y_j}{1 - l_{ii}},$$

  where the $1 - l_{ii}$ term in the denominator renormalizes the linear weights so that they still sum to 1

Closed form for cross-validation

- This leads to the further simplification:

$$\frac{1}{n}\sum_i \left\{y_i - \hat{f}_{(-i)}(x_i)\right\}^2 = \frac{1}{n}\sum_i \left(\frac{y_i - \hat{y}_i}{1 - l_{ii}}\right)^2$$

- Thus, we can obtain a closed form solution for the leave-one-out cross-validation score from a single fit
- **Homework:** Prove that the above relationship holds for any linear smoother such that $\hat{f}_{(-i)}(x_i)$ can be obtained by the expression on the previous slide.

Generalized cross-validation

- Computing $\mathbf{L}$, however, would require matrix inversion
- As with regular linear regression, this is an inefficient way to solve for $\hat{\beta}$ and $\hat{\mathbf{y}}$
- However, $\mathrm{tr}(\mathbf{L})$ can still be obtained easily
- This is the motivation behind *generalized cross-validation*:

$$GCV = \frac{1}{n} \sum_i \left(\frac{y_i - \hat{y}_i}{1 - \mathrm{tr}(\mathbf{L})/n} \right)^2$$

Generalized cross-validation (cont'd)

- GCV is equal to CV if all the l_{ii}'s are equal; otherwise, they will be different, although often quite close
- Note that, for $x \approx 0$, $1/(1-x)^2 \approx 1 + 2x$; thus,

$$GCV \approx \frac{1}{n}\sum_i (y_i - \hat{y}_i)^2 + \frac{2\hat{\sigma}^2 \mathrm{tr}(\mathbf{L})}{n},$$

  where $\hat{\sigma}^2 = n^{-1}\sum_i (y_i - \hat{y}_i)^2$
- If we multiply by n and divide by $\hat{\sigma}^2$, we have that GCV is approximately proportional to $-2\mathrm{loglik} + 2\mathrm{tr}(\mathbf{L})$, the AIC of the fit
- Note that, in this approximation, $\mathrm{tr}(\mathbf{L})$ acts as the *effective degrees of freedom* in the fit

Effective degrees of freedom

- Note that the smoothing matrix is quite similar to the *projection matrix* or *hat matrix* from linear regression ($\mathbf{H} = \mathbf{X}(\mathbf{X}'\mathbf{X})^{-1}\mathbf{X}'$), for which $\hat{\mathbf{y}} = \mathbf{H}\mathbf{y}$
- Both $\mathbf{L}$ and $\mathbf{H}$ are symmetric and positive semidefinite
- However, $\mathbf{H}$ is idempotent (*i.e.*, $\mathbf{HH} = \mathbf{H}$), whereas $\mathbf{LL}$ is smaller than $\mathbf{L}$ (in the sense that $\mathbf{L} - \mathbf{LL}$ is positive semidefinite), because $\mathbf{L}$ introduces *shrinkage*, biasing estimates towards zero in order to reduce variance
- In linear regression, $\mathrm{tr}(\mathbf{H})$ is equal to the degrees of freedom; for linear smoothers, $\mathrm{tr}(\mathbf{H})$ defines the effective degrees of freedom (this analogy can be made more rigorous using eigenvalues)

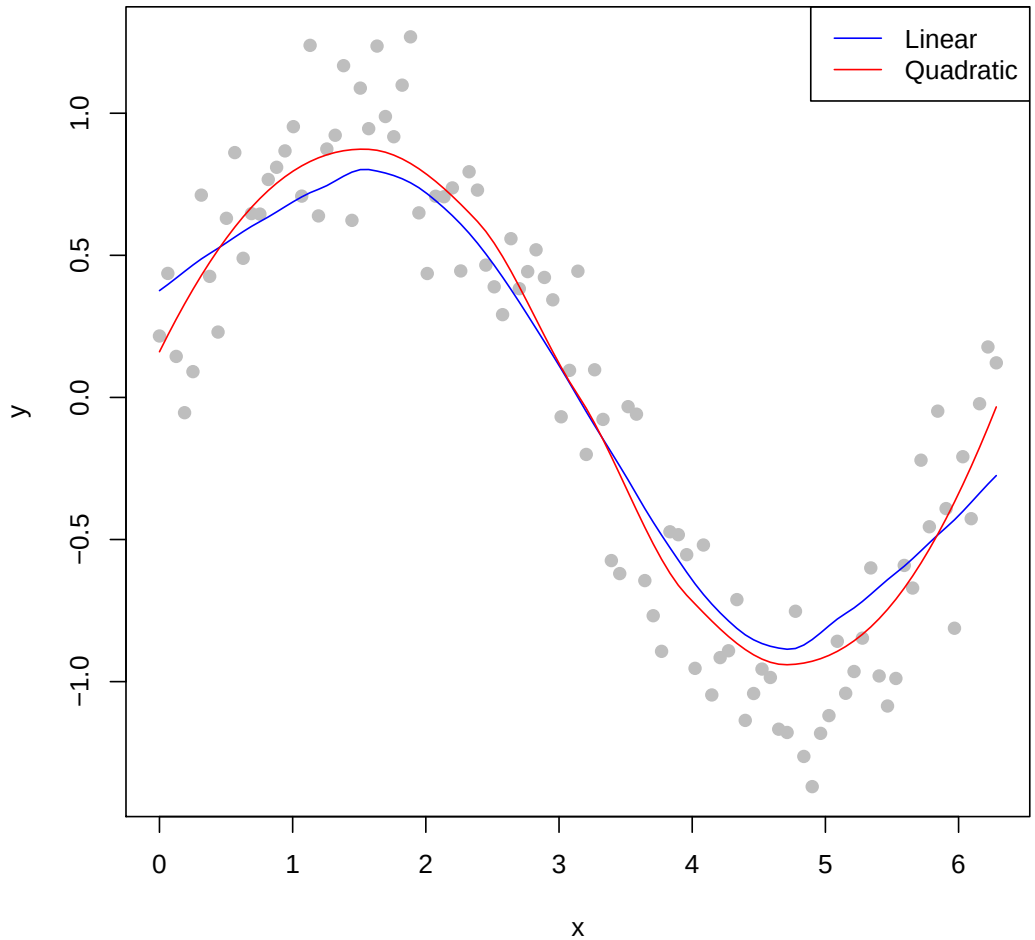
The loess function

- In R, local linear regression is implemented through the loess function, which uses a formula interface similar to that of other regression functions:

```
fit <- loess(spnbmd~age,bmd.data,span=0.3,degree=1)
```

- The two key options are
  - span: this is the smoothing parameter which controls the bias-variance tradeoff
  - degree: this lets you specify local constant regression (the Nadaraya-Watson estimator from earlier, degree=0), local linear regression (degree=0), or local polynomial fits (degree=2)

Bias due to local linear fitting



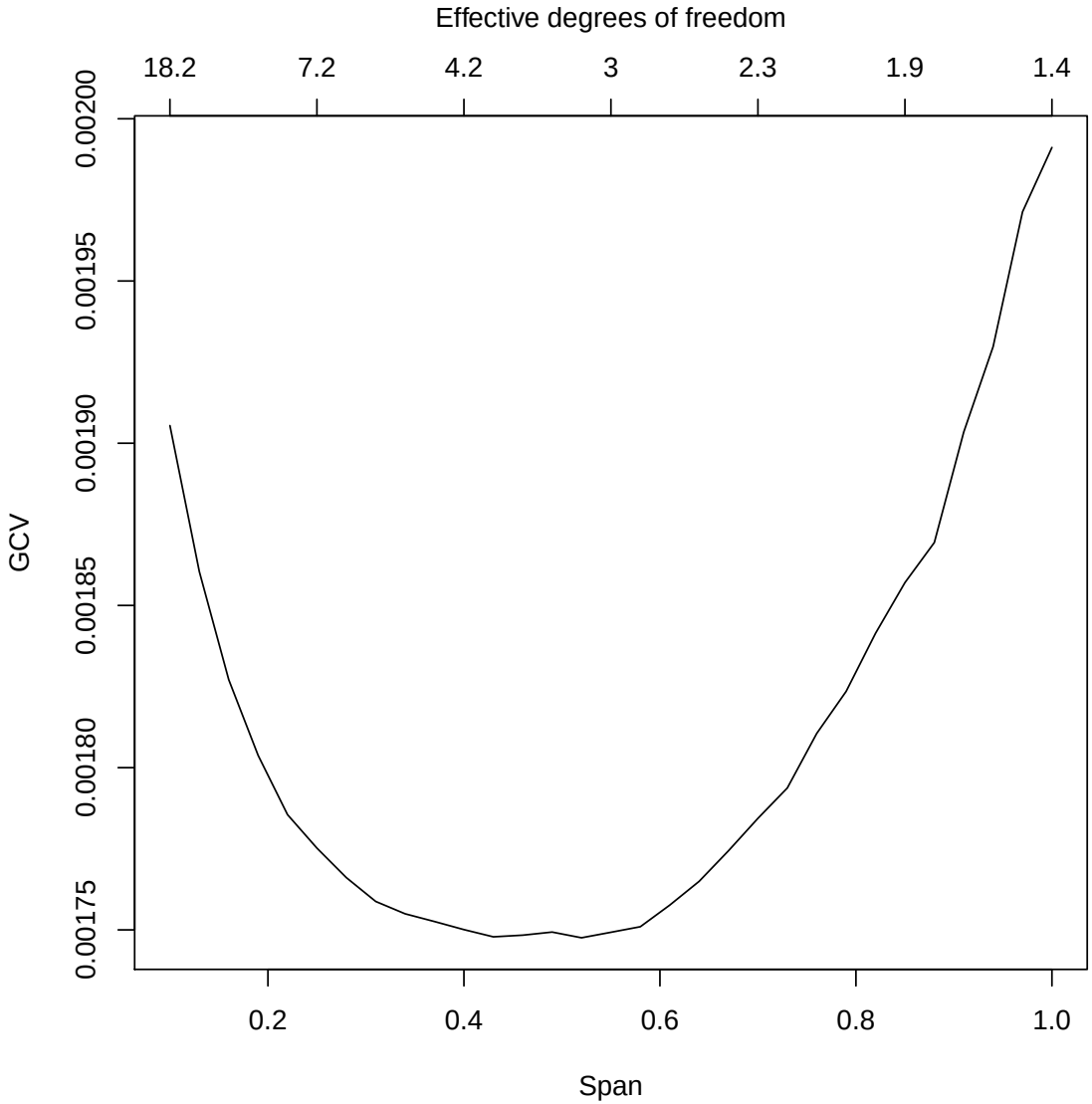
Local linear versus local quadratic fitting

- As the figure on the previous slide indicates, local linear models tend to be biased in regions of high curvature, a phenomenon referred to as "trimming the hills and filling in the valleys"
- Higher-order local polynomials correct for this bias, but at the expense of increased variability
- The conventional wisdom on the subject of local linear versus local quadratic fitting says that:
  - Local linear fits tend to be superior at the boundaries
  - Local quadratic fits tend to be superior in the interior
  - Local fitting to higher order polynomials is possible in principle, but rarely necessary in practice

The span argument

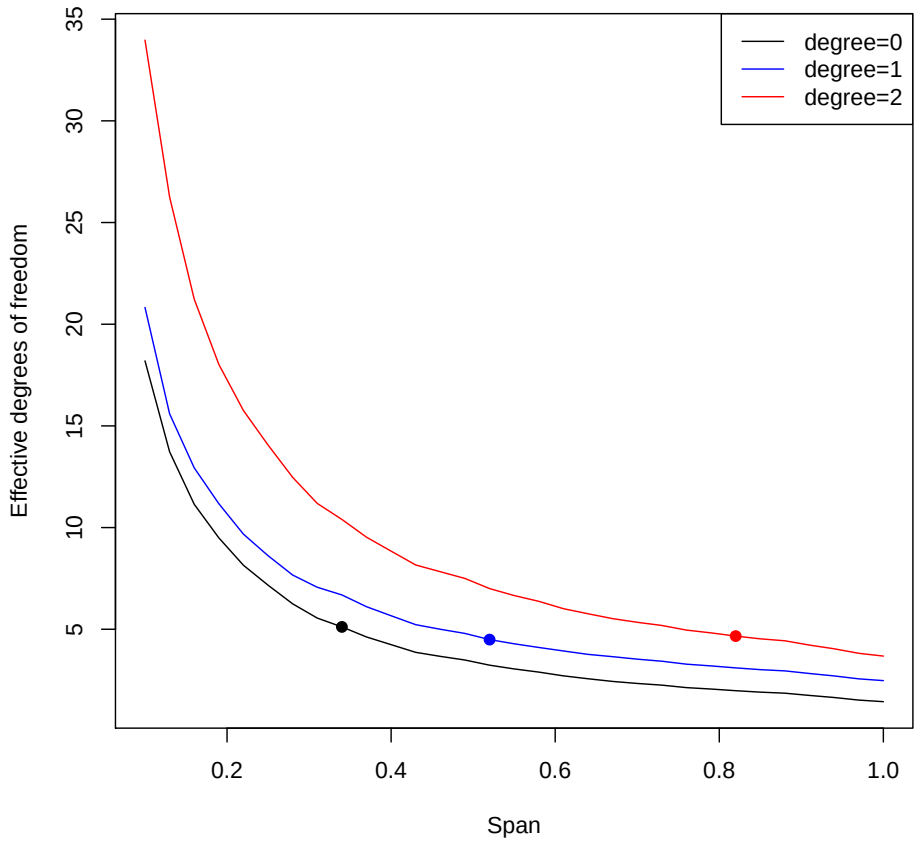
- The other important option in `span`, which controls the degree of smoothing
- Unlike `density`, `loess` does not allow you to choose your own kernel; only the tricube kernel is implemented, and `span` refers to the proportion of the observations $\{x_i\}$ within its compact support
- Also unlike `density`, the kernel in `loess` is adaptive
- Thus, specifying `span=0.2` means that the bandwidth of the kernel at x_0 is made just wide enough to include 20% of the x_i values

Selection of smoothing parameter

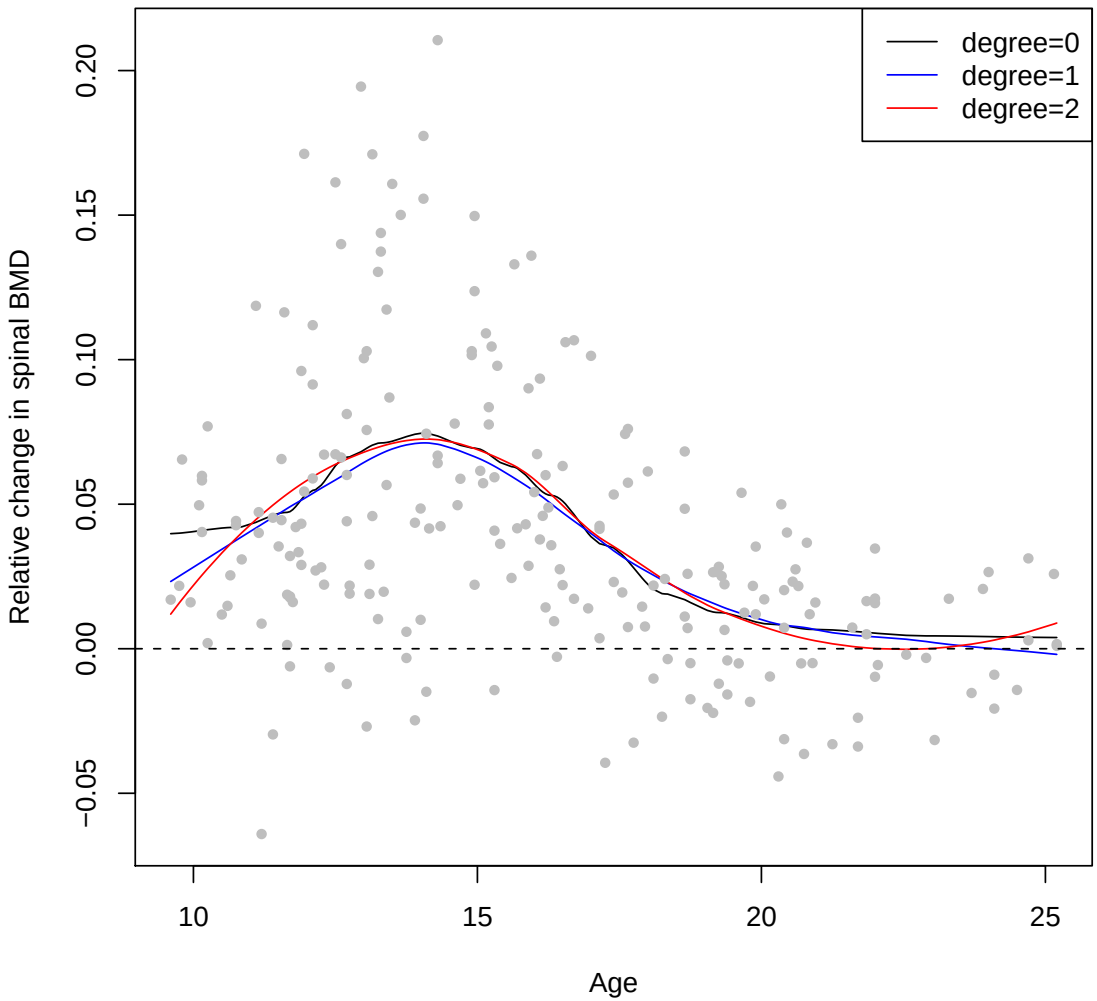


Effective degrees of freedom versus span

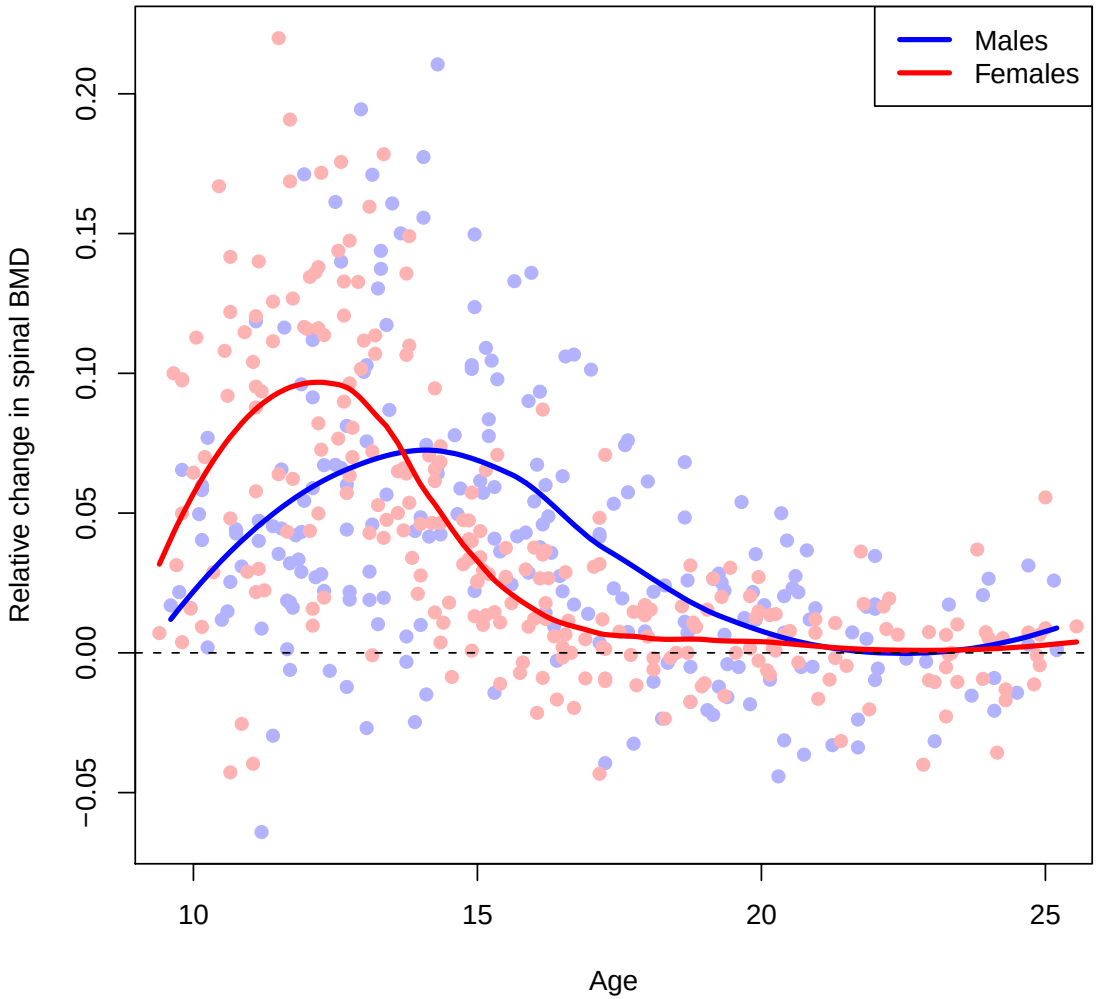
Dots indicate optimal smoothing, as chosen by GCV:



Optimal fits for the bone mineral density data



Bone mineral density data – males versus females



Pointwise inference

- At any given target point x_0, $\hat{f}$ is a simple linear model
- Thus,

$$\mathbb{E}\hat{f}(x_0) = \sum_i l_i(x_0) f(x_0)$$

$$\mathbb{V}\hat{f}(x_0) = \sigma^2 \sum_i l_i(x_0)^2,$$

  where $\sigma^2 = \mathbb{V}(y)$

- One method of constructing pointwise confidence intervals, then, is via

$$\hat{f}(x_0) \pm z_{\alpha/2}\sigma^2 \sum_i l_i(x_0)^2,$$

The bias problem

- However, note once again that $\hat{f}$ is not an unbiased estimate of f:

$$\frac{\hat{f} - f}{SD(\hat{f})} \sim Z + \frac{\text{bias}}{\sqrt{\text{variance}}},$$

  where $Z \sim N(0, 1)$
- Thus, usual normal-theory methods for confidence intervals will result in confidence intervals for $\bar{f} = \mathbb{E}(\hat{f})$, not for f itself
- However, this is generally ignored: the vast majority of confidence intervals in nonparametric regression just accept the fact that the interval is technically an interval for $\bar{f}$, not f

Estimation of σ^2

- Note that

$$\mathbb{E}\sum_i (y_i - \hat{y}_i)^2 = \sigma^2 \mathrm{tr}\left((\mathbf{I}-\mathbf{L})'(\mathbf{I}-\mathbf{L})\right) + \mathbf{b}'\mathbf{b},$$

  where $\mathbf{b} = \mathbb{E}(\mathbf{y}) - \mathbb{E}(\hat{\mathbf{y}})$
- Thus, if n is reasonably large with respect to the effective degrees of freedom, the second term will be close to zero, and the following is a nearly unbiased estimator for σ^2:

$$\hat{\sigma}^2 = \frac{\sum_i (y_i - \hat{y}_i)^2}{n - 2\nu + \tilde{\nu}},$$

  where $\nu = \mathrm{tr}(\mathbf{L})$ and $\tilde{\nu} = \mathrm{tr}(\mathbf{L}'\mathbf{L})$
- The quantity $2\nu - \tilde{\nu}$ is known as the *equivalent number of parameters*, by analogy with linear regression

Global confidence bands

- It is also possible to obtain simultaneous confidence bands across the entire range of x
- The details are fairly complicated and rest on representing

$$W(x) = \frac{\hat{f}(x) - \bar{f}(x)}{\sigma \|l(x)\|}$$

  as a Gaussian process, where $\|l(x)\| = \sqrt{\sum_i l_i(x)}$
- We won't go into the details (they are in Section 5.7 of our text), but we will talk about how to obtain confidence bands using the `locfit` package in `R`

The `locfit` function

- The basic syntax of model fitting is as follows:

  ```
  fit <- locfit(spnbmd~lp(age,nn=.7,deg=2))
  ```

  where `lp` controls the local polynomial which is fit to the data
- Just like `loess`, there is a `nn` parameter (analogous to `span`), which adaptively determines the bandwidth by setting the number of points in the neighborhood of x_0 equal to `nn`
- There is also a `deg` parameter, which controls the degree of the local polynomial (like `loess`, the default is 2)

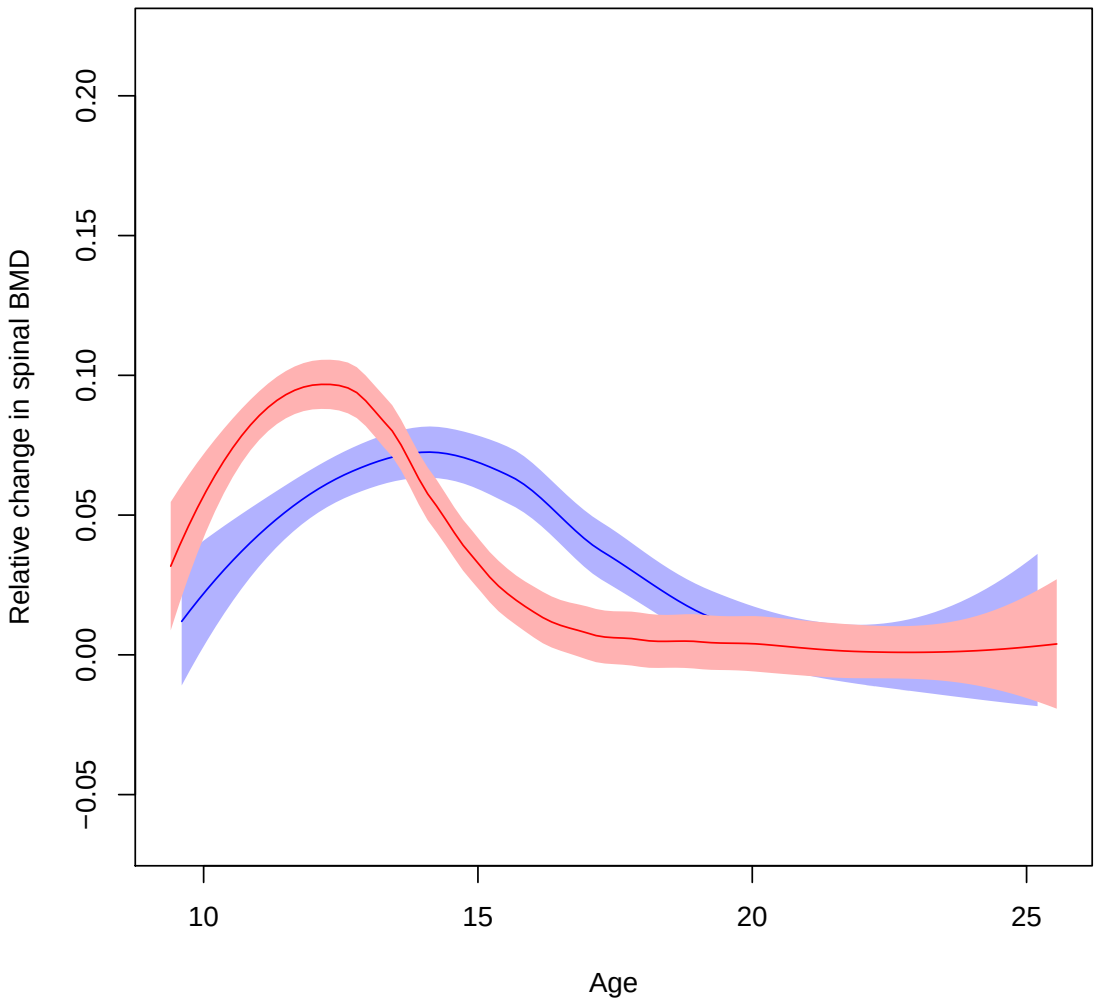
Confidence intervals in `locfit`

- The `locfit` package is very well-developed, and we cannot possibly cover all of its features here
- However, two very important features are its ability to construct pointwise and simultaneous confidence bands:

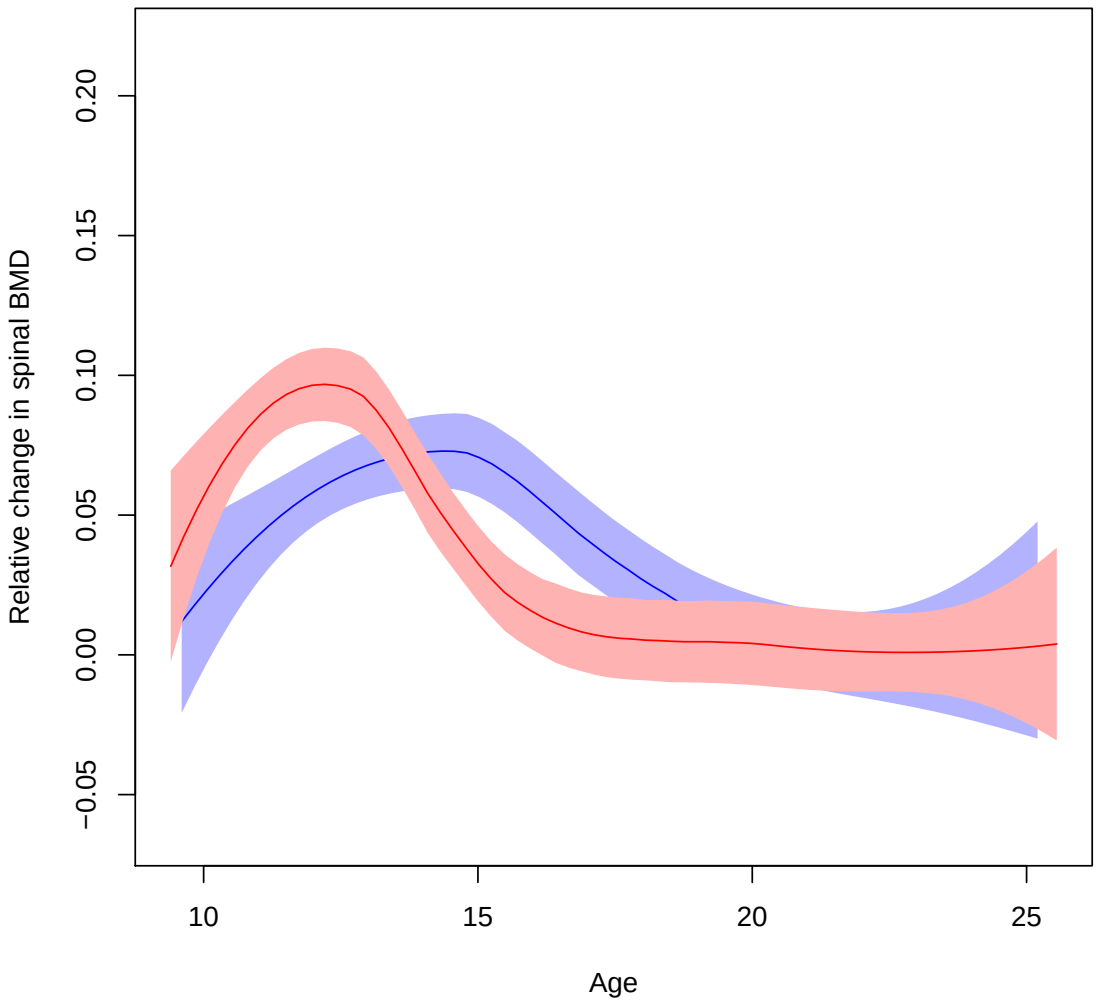
```
predict(fit,newdata=seq(9,25,len=75),se=TRUE)
scb(spnbmd~lp(age)) # Simultaneous conf. bands
plot(fit,band="global") # Plot the band
```

where the first line of code returns the value of $\hat{f}(\mathbf{x}_0)$ and its standard error for a supplied list of points `newdata`

Pointwise CIs for the bone mineral density data



Simultaneous CIs for the bone mineral density data



Asymptotic expected loss

- Suppose that the x_i's are drawn from a distribution with density $g(x)$
- Also, suppose that both g and f (the regression function) are continuous at a point x_0, with $g(x) > 0$
- Finally, suppose that $h \to 0$ and $nh \to \infty$ as $n \to \infty$
- **Theorem:** Both the local linear regression estimator and the Nadaraya-Watson kernel estimator have variance

$$\frac{\sigma^2}{g(x)nh} \int K^2(u)du + o_p\left(\frac{1}{nh}\right)$$

Asymptotic expected loss (cont'd)

Theorem (cont'd): However, the Nadaraya-Watson estimator has bias

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

whereas the local linear regression estimator has bias

$$h^2 \frac{1}{2} f''(x) \int u^2 K(u) du + o_p(h^2)$$

Comments

- Bias that depends on the distribution of the distribution of x is called *design bias*; note that the local linear estimator is free of design bias, whereas the kernel average is not
- Note also that the local linear estimator is free of bias to first order
- These results hold more generally, with local quadratic regression eliminating bias through second order (*i.e.* the leading bias term has a $f'''(x)$ in it)