Splines

Patrick Breheny

November 20

Introduction

- We are discussing ways to estimate the regression function f, where

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

- One approach is of course to assume that f has a certain shape, such as linear or quadratic, that can be estimated parametrically
- We have also discussed locally weighted linear/polynomial models as a way of allowing f to be more flexible
- An alternative approach is to introduce *local basis functions*

Basis functions

- A common approach for extending the linear model is to augment the linear component of x with additional, known functions of x:

$$f(x) = \sum_{m=1}^{M} \beta_m h_m(x),$$

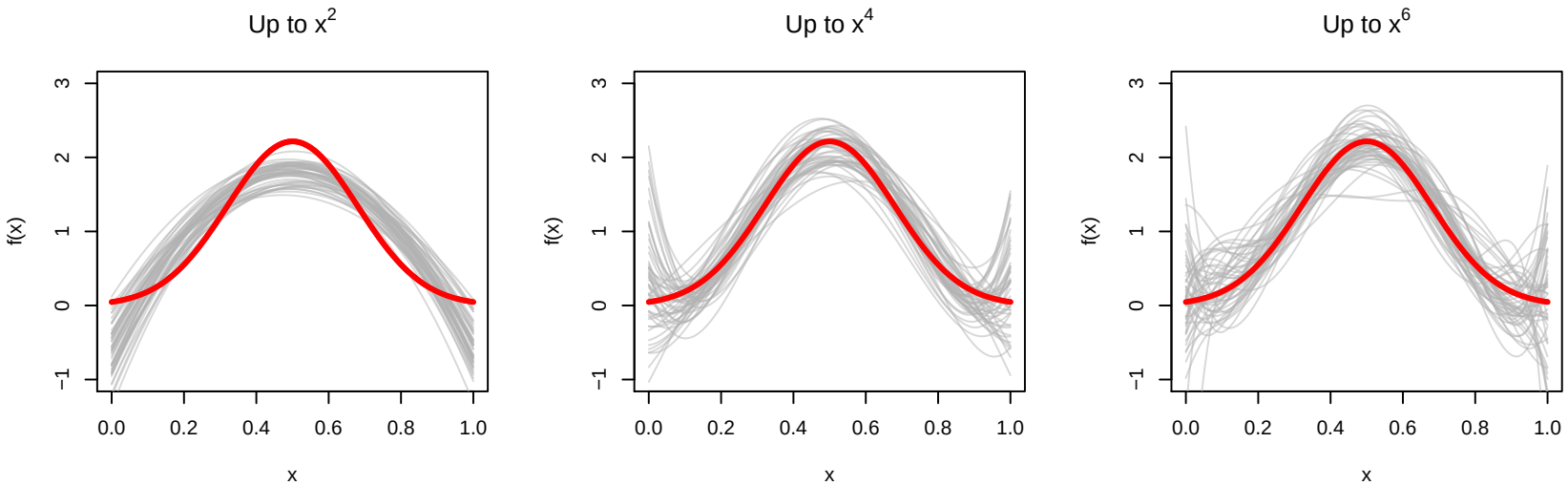
  where the $\{h_m\}$ are known functions called *basis functions*
- Because the basis functions $\{h_m\}$ are prespecified and the model is linear in these new variables, ordinary least squares approaches for model fitting and inference can be employed
- This idea is not new to you, as you have encountered transformations and the inclusion of polynomial terms in models in earlier courses

Problems with polynomial regression

- However, polynomial terms introduce undesirable side effects: each observation affects the entire curve, even for x values far from the observation
- Not only does this introduce bias, but it also results in extremely high variance near the edges of the range of x
- As Hastie *et al.* (2009) put it, "tweaking the coefficients to achieve a functional form in one region can cause the function to flap about madly in remote regions"

Problems with polynomial regression (cont'd)

To illustrate this, consider the following simulated example (gray lines are models fit to 100 observations arising from the true f, colored red):



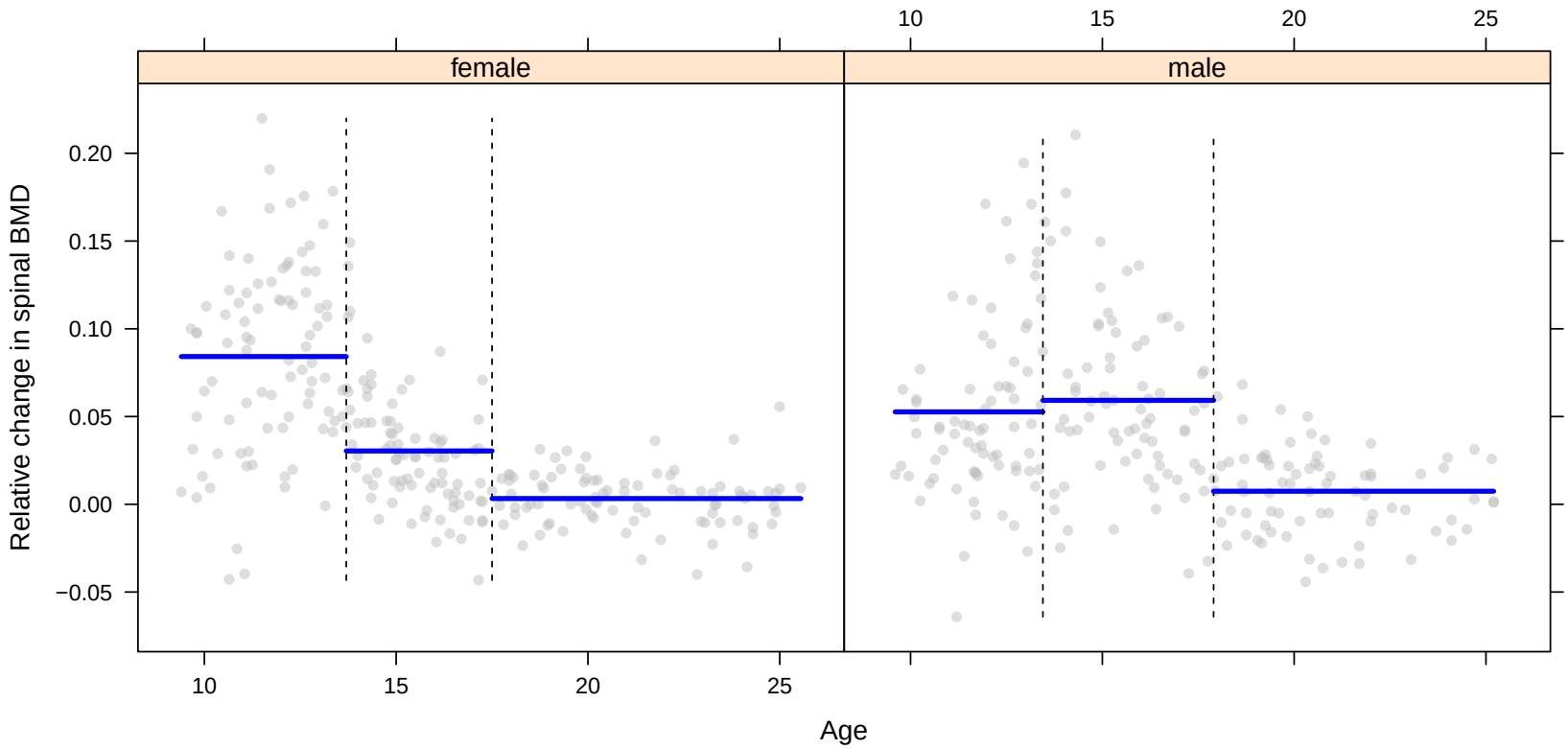
Global versus local bases

- We can do better than this
- Let us consider instead *local* basis functions, thereby ensuring that a given observation affects only the nearby fit, not the fit of the entire line
- In this lecture, we will discuss *splines*: piecewise polynomials joined together to make a single smooth curve

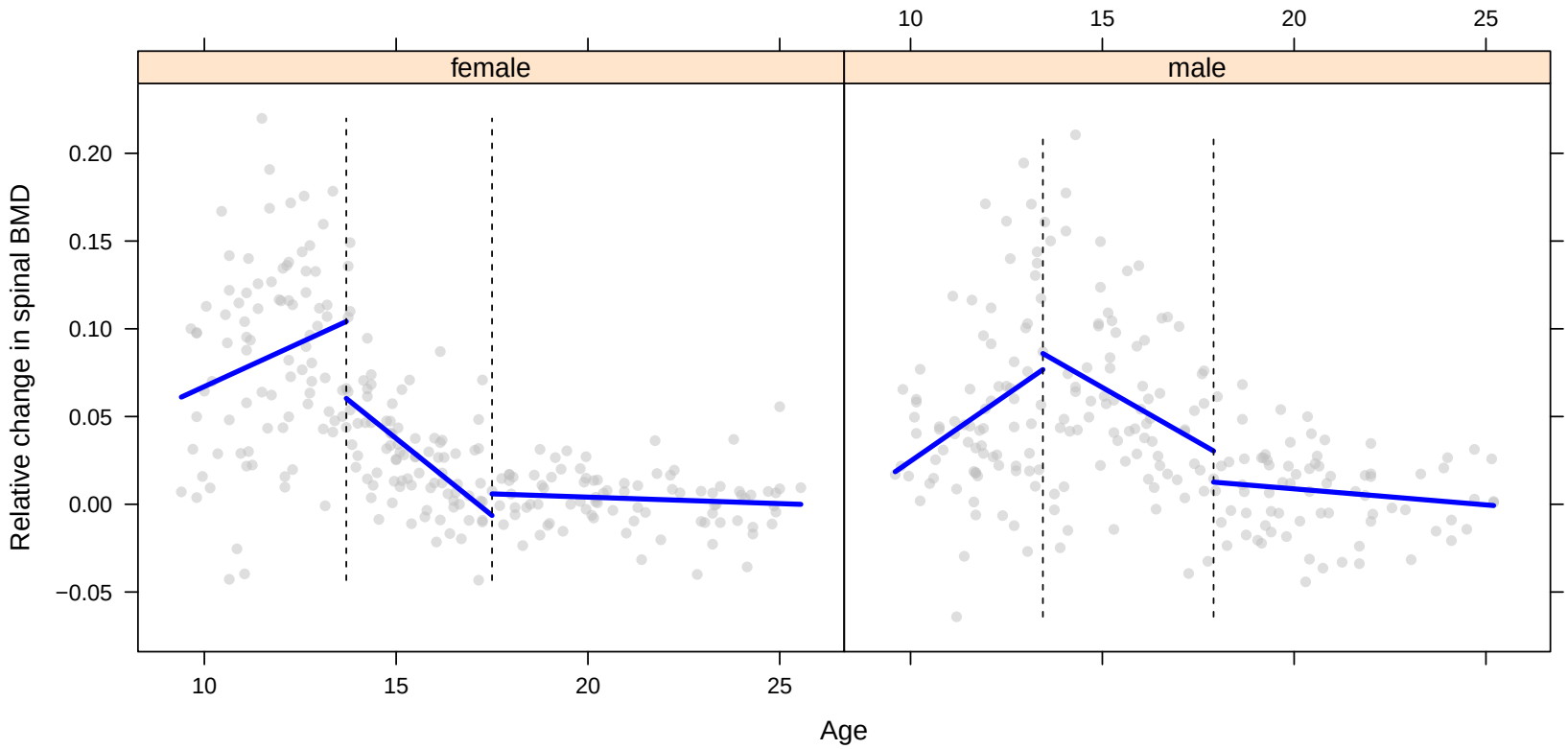
The piecewise constant model

- To understand splines, we will gradually build up a piecewise model, starting at the simplest one: the piecewise constant model
- First, we partition the range of x into $K+1$ intervals by choosing K points $\{\xi_k\}_{k=1}^K$ called *knots*
- For our example involving bone mineral density, we will choose the tertiles of the observed ages

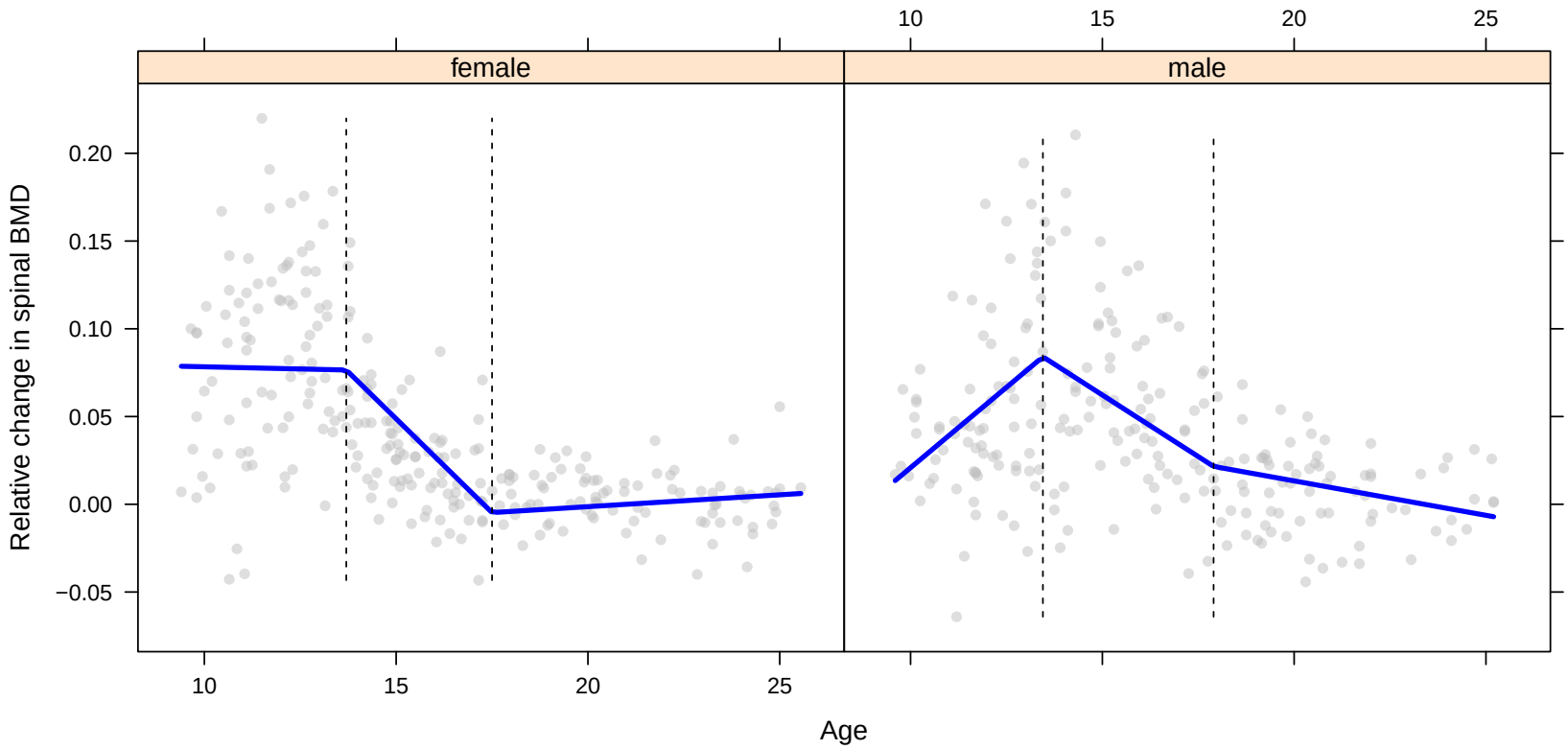
The piecewise constant model (cont'd)



The piecewise linear model



The continuous piecewise linear model



Basis functions for piecewise continuous models

These constraints can be incorporated directly into the basis functions:

$$h_1(x) = 1, \quad h_2(x) = x, \quad h_3(x) = (x - \xi_1)_+, \quad h_4(x) = (x - \xi_2)_+,$$

where $(\cdot)_+$ denotes the positive portion of its argument:

$$r_+ = \begin{cases} r & \text{if } r \geq 0 \\ 0 & \text{if } r < 0 \end{cases}$$

Basis functions for piecewise continuous models

- It can be easily checked that these basis functions lead to a composite function $f(x)$ that:
  - Is linear everywhere except the knots
  - Has a different intercept and slope in each region
  - Is everywhere continuous
- Also, note that the degrees of freedom add up: 3 regions $\times$ 2 degrees of freedom in each region - 2 constraints $=$ 4 basis functions

Splines

- The preceding is an example of a *spline*: a piecewise $m-1$ degree polynomial that is continuous up to its first $m-2$ derivatives
- By requiring continuous derivatives, we ensure that the resulting function is as smooth as possible
- We can obtain more flexible curves by increasing the degree of the spline and/or by adding knots
- However, there is a tradeoff:
  - Few knots/low degree: Resulting class of functions may be too restrictive (bias)
  - Many knots/high degree: We run the risk of overfitting (variance)

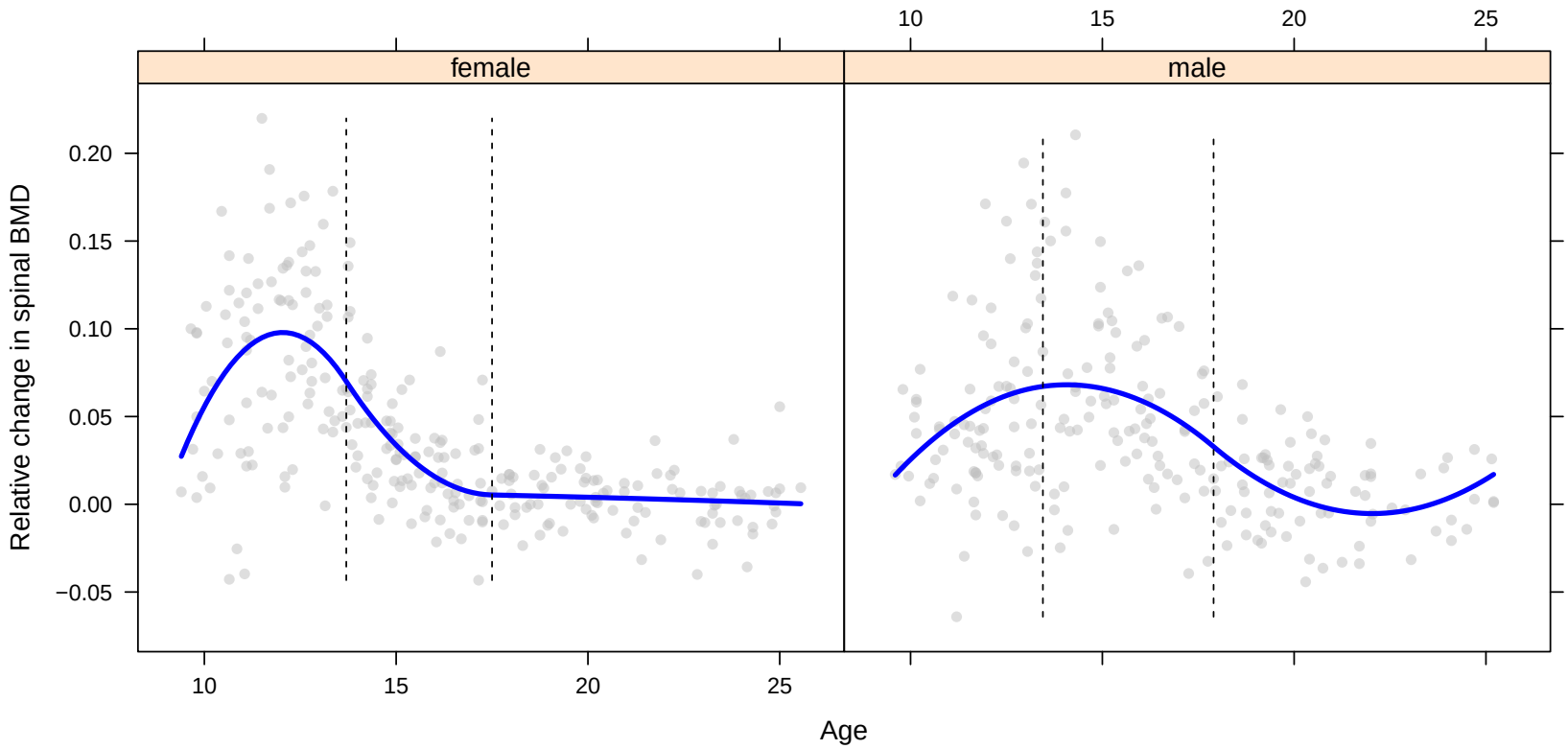
The truncated power basis

- The set of basis functions introduced earlier is an example of what is called the *truncated power basis*
- Its logic is easily extended to splines of order m:

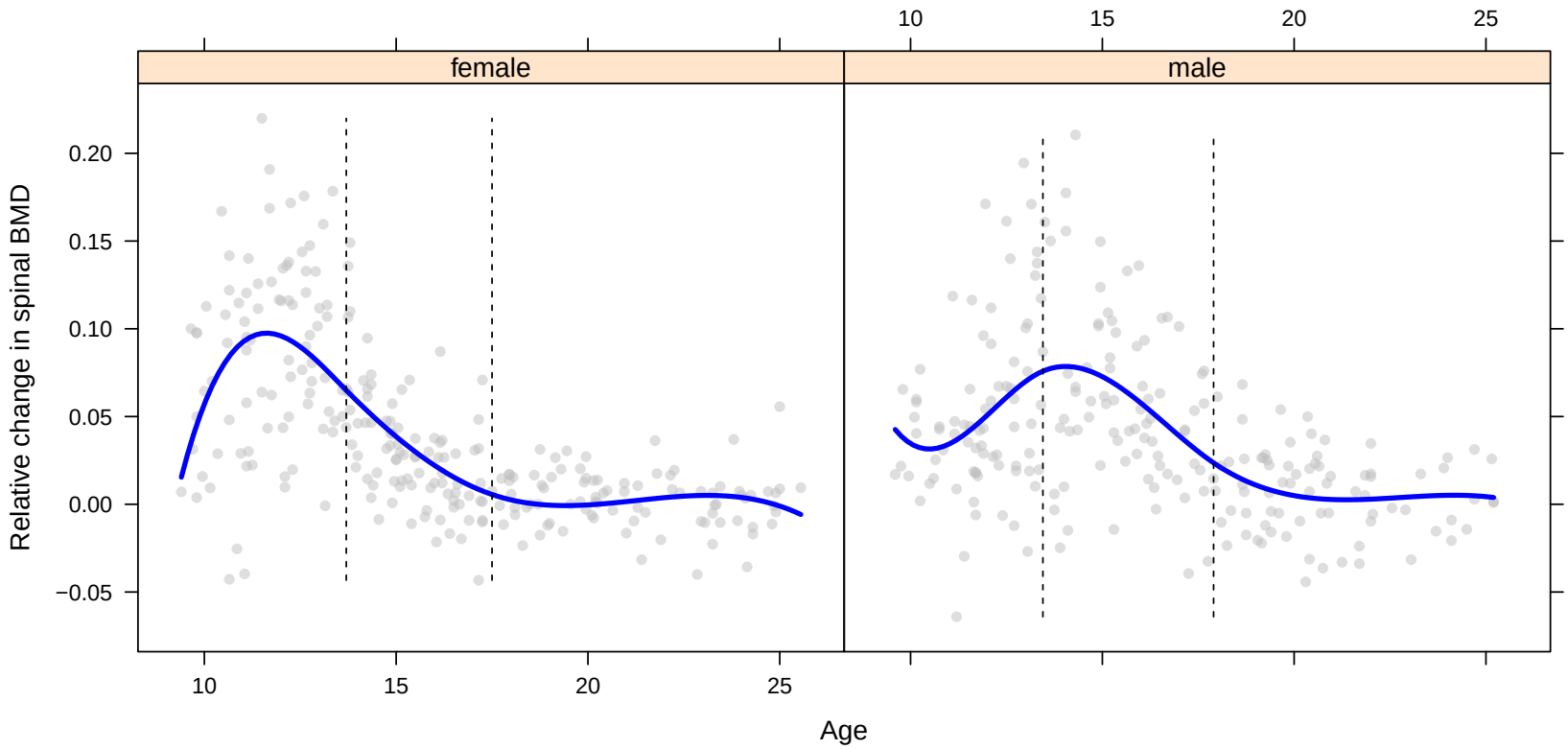
$$h_j(x) = x^{j-1} \qquad j = 1, \ldots, m$$
$$h_{m+k}(x) = (x - \xi_k)_+^{m-1} \qquad k = 1, \ldots, K$$

- Note that a spline has $m + K$ degrees of freedom
- **Homework:** Write the set of truncated spline basis functions for representing a cubic spline function with three knots.

Quadratic splines



Cubic splines



Additional notes

- These types of fixed-knot models are referred to as *regression splines*
- Recall that cubic splines contain $4 + K$ degrees of freedom: $K + 1$ regions $\times$ 4 parameters per region - K knots $\times$ 3 constraints per knot
- It is claimed that cubic splines are the lowest order spline for which the discontinuity at the knots cannot be noticed by the human eye
- There is rarely any need to go beyond cubic splines, which are by far the most common type of splines in practice

Implementing regression splines

- The truncated power basis has two principal virtues:
  - Conceptual simplicity
  - The linear model is nested inside it, leading to simple tests of the null hypothesis of linearity
- Unfortunately, it has several computational/numerical flaws – it's inefficient and can lead to overflow and nearly singular matrix problems
- The more complicated but numerically much more stable and efficient *B-spline* basis is often employed instead

B-splines in R

- Fortunately, one can use B-splines without knowing the details behind their complicated construction
- In the `splines` package (which by default is installed but not loaded), the `bs()` function will implement a B-spline basis for you

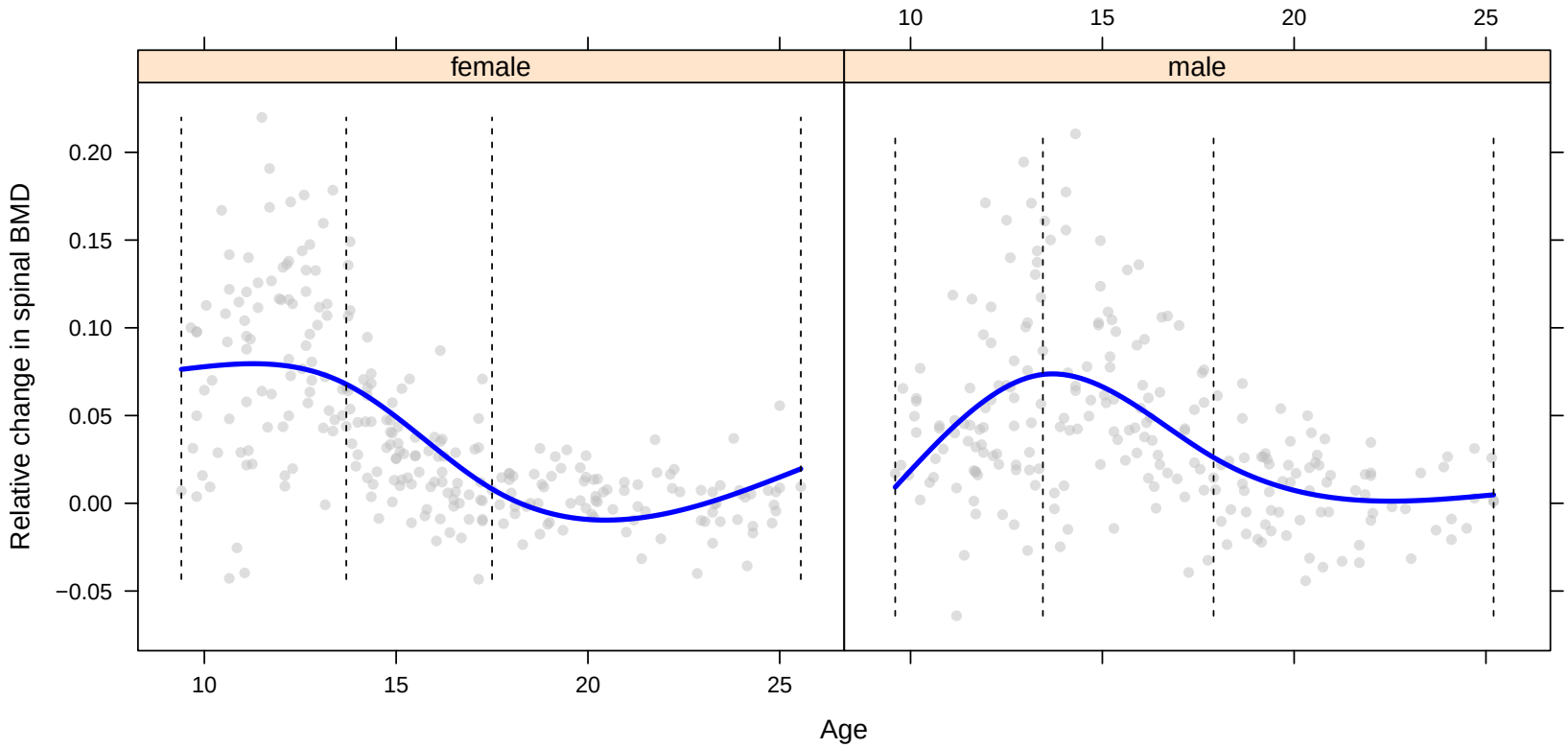
```
X <- bs(x,knots=quantile(x,p=c(1/3,2/3)))
X <- bs(x,df=5)
X <- bs(x,degree=2,df=10)
Xp <- predict(X,newdata=x)
```

- By default, `bs` uses `degree=3`, knots at evenly spaced quantiles, and does not return a column for the intercept

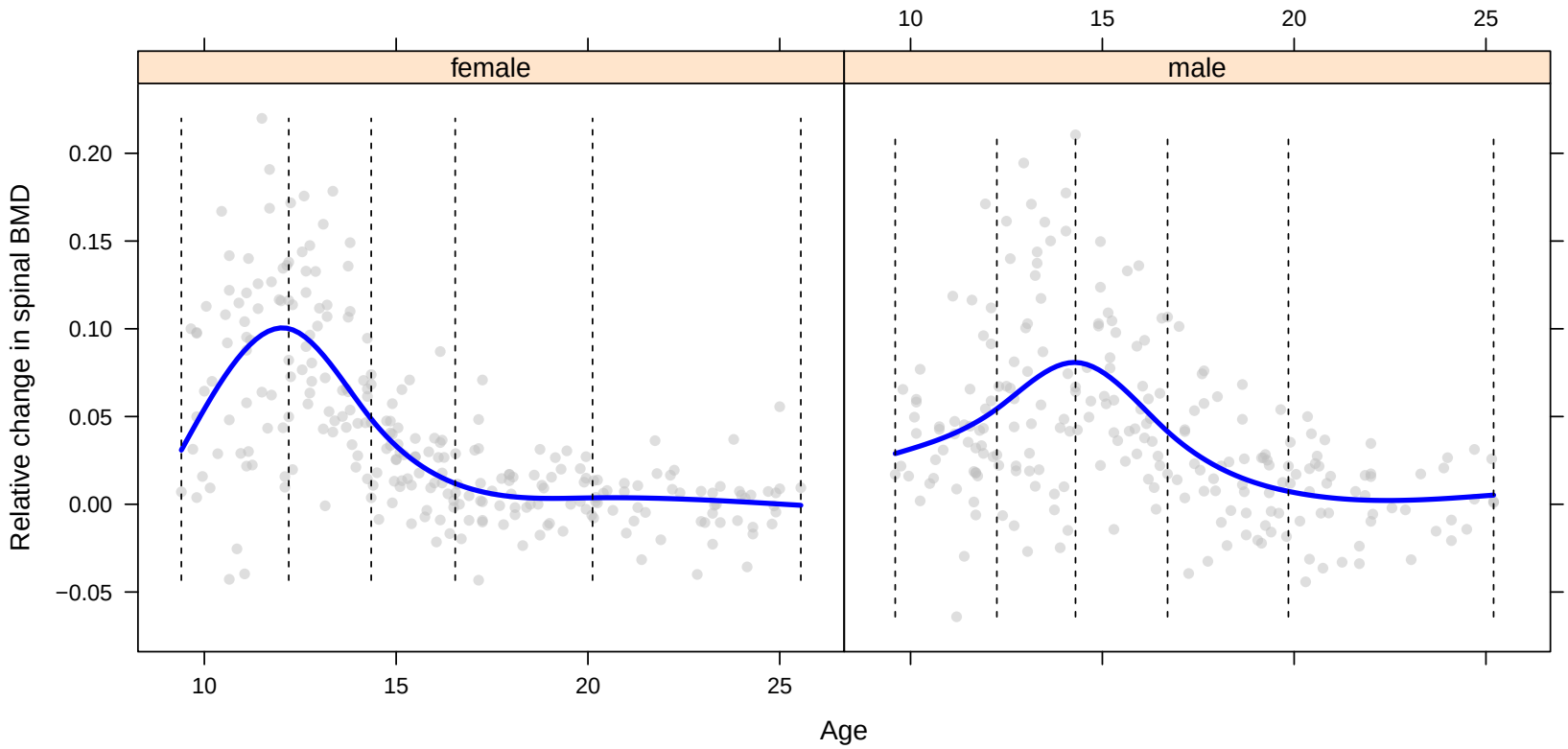
Natural cubic splines

- Polynomial fits tend to be erratic at the boundaries of the data; naturally, cubic splines share the same flaw
- *Natural cubic splines* ameliorate this problem by adding the additional (4) constraints that the function is linear beyond the boundaries of the data

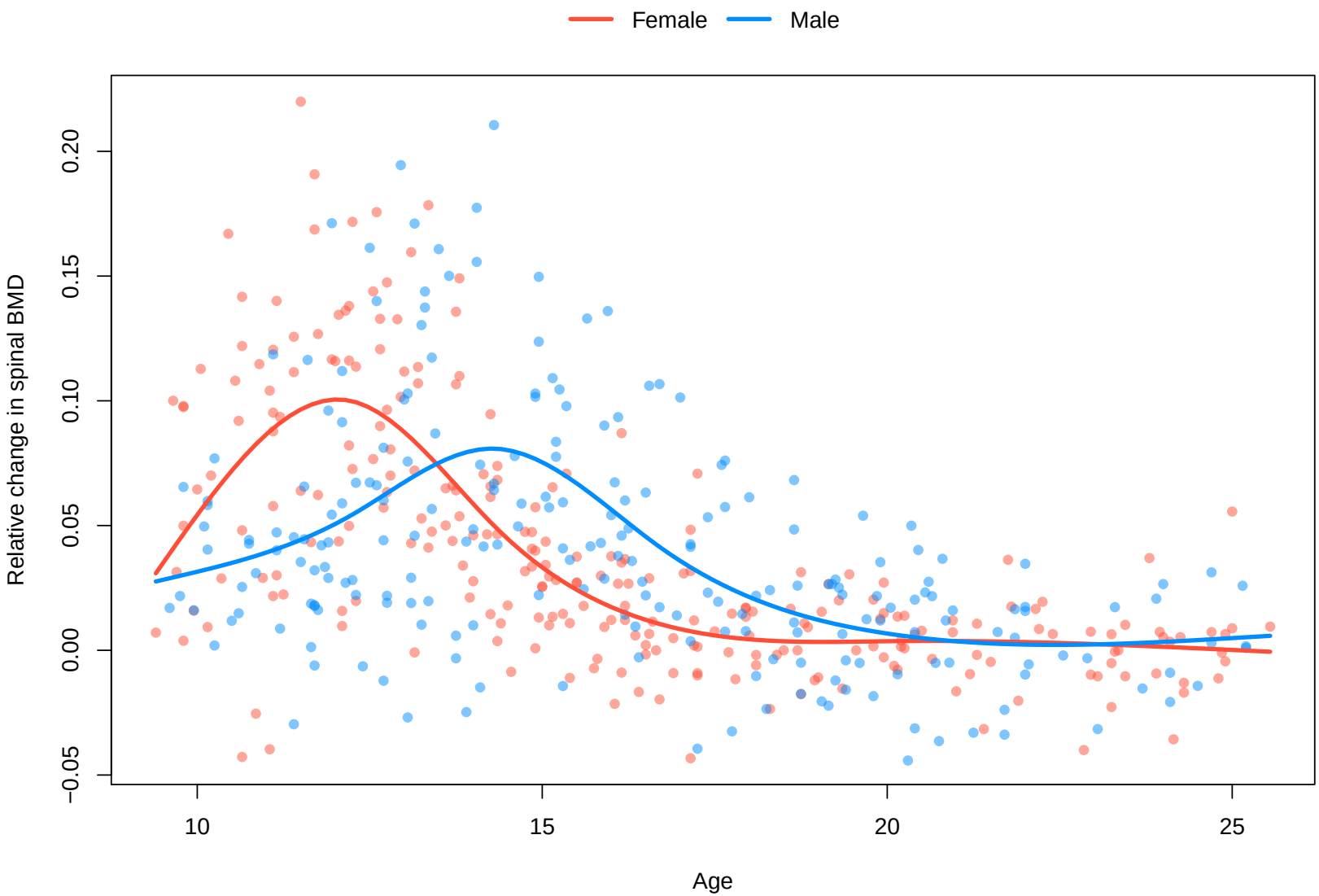
Natural cubic splines (cont'd)



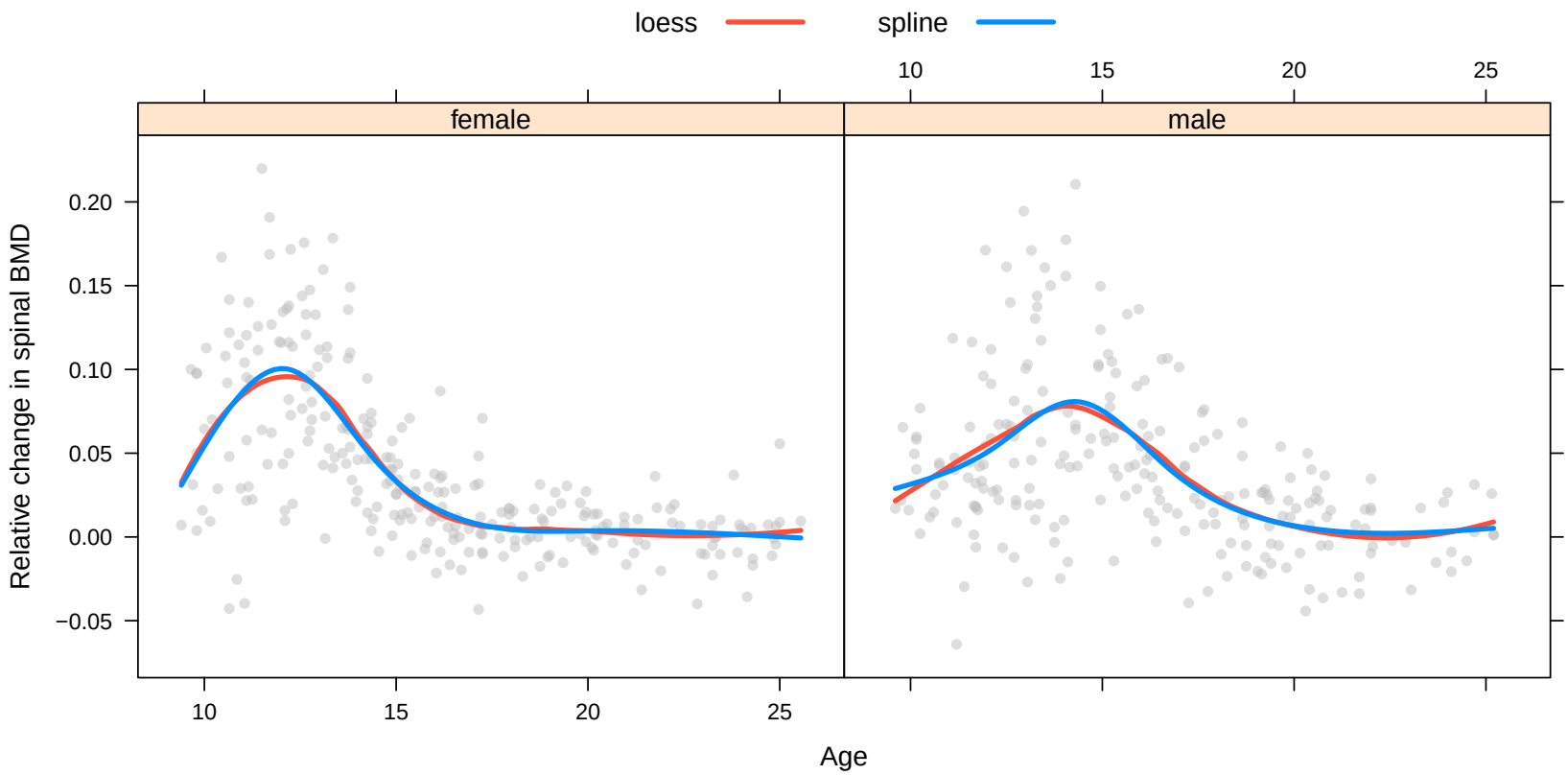
Natural cubic splines, 6 df



Natural cubic splines, 6 df (cont'd)



Splines vs. Loess (6 df each)



Natural splines in R

- R also provides a function to compute a basis for the natural cubic splines, ns, which works almost exactly like bs, except that there is no option to change the degree
- Note that a natural spline has $m + K - 4$ degrees of freedom; thus, a natural cubic spline with K knots has K degrees of freedom

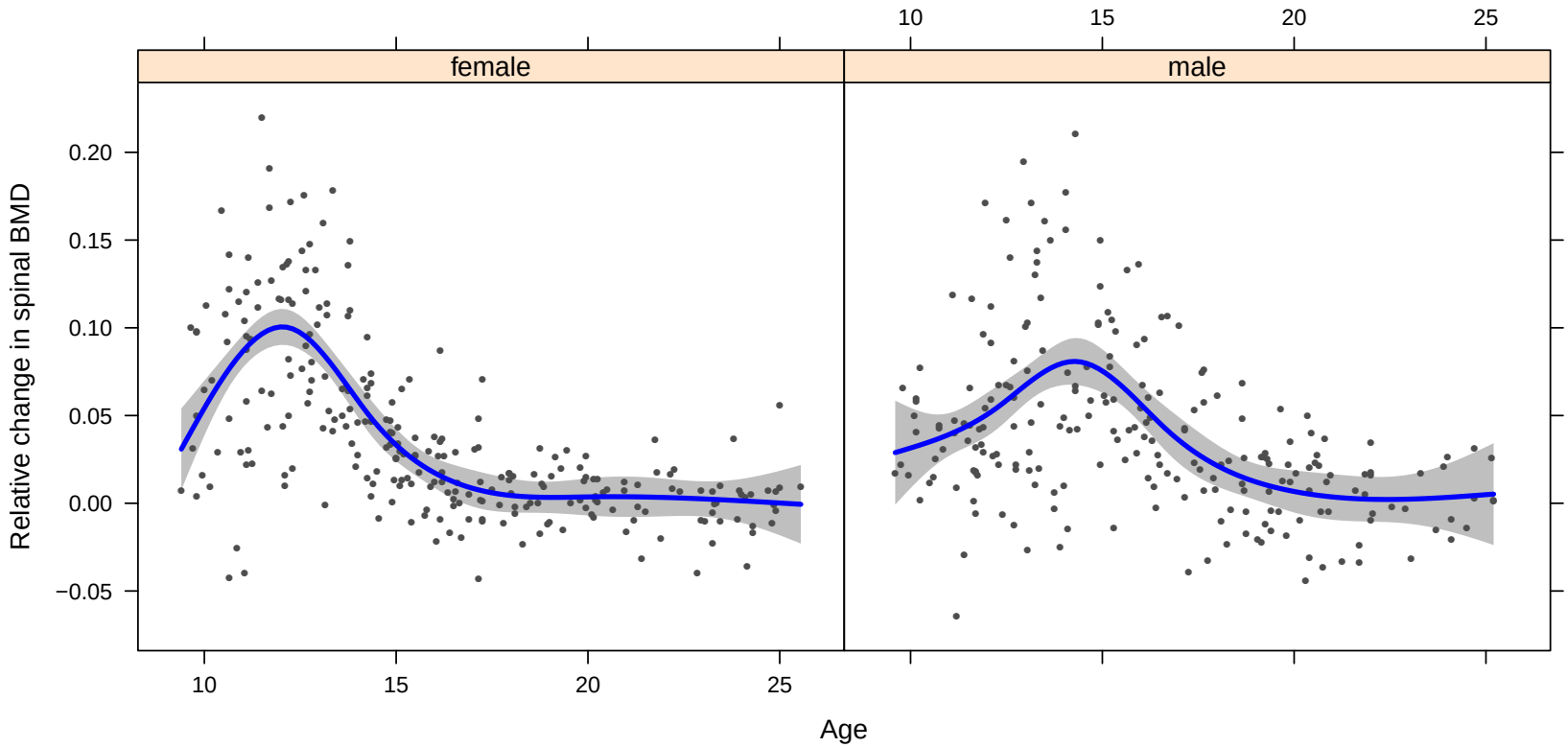
Mean and variance estimation

- Because the basis functions are fixed, all standard approaches to inference for regression are valid
- Furthermore, note that the resulting estimate is a linear smoother with $\mathbf{L} = \mathbf{X}(\mathbf{X}'\mathbf{X})^{-1}\mathbf{X}'$; thus

$$\mathbb{E}(\hat{\mathbf{f}}) = \mathbf{L}\mathbf{f} \quad \text{where } \mathbf{f} = \mathbb{E}(\mathbf{y}|\mathbf{x})$$
$$\mathbb{V}(\hat{\mathbf{f}}) = \sigma^2\mathbf{L}\mathbf{L}'$$
$$CV = \frac{1}{n}\sum_i \left(\frac{y_i - \hat{y}_i}{1 - l_{ii}}\right)^2$$

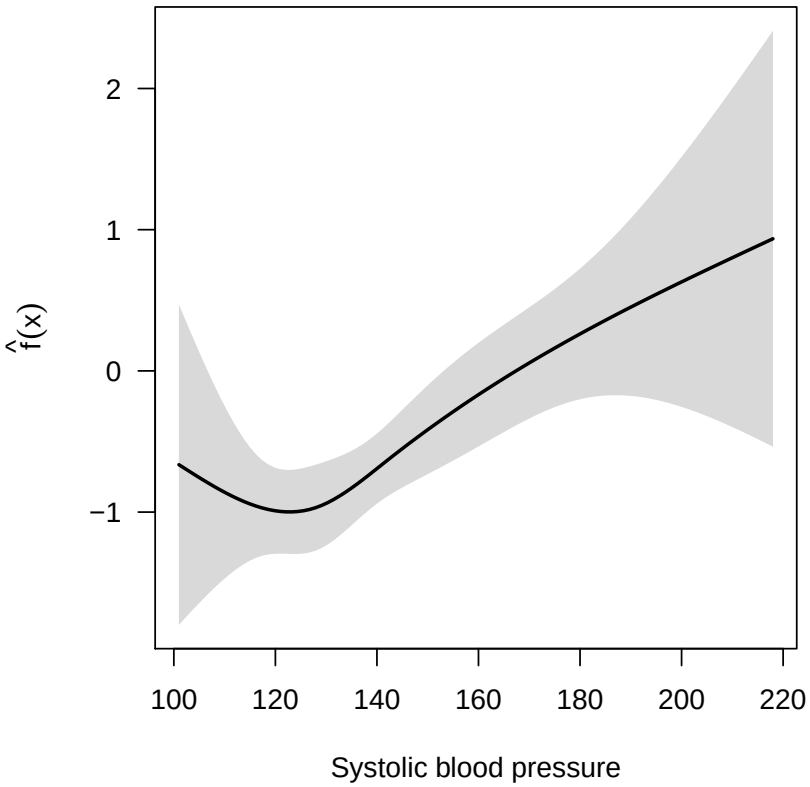
- Furthermore, extensions to logistic regression, Cox proportional hazards regression, etc., are straightforward

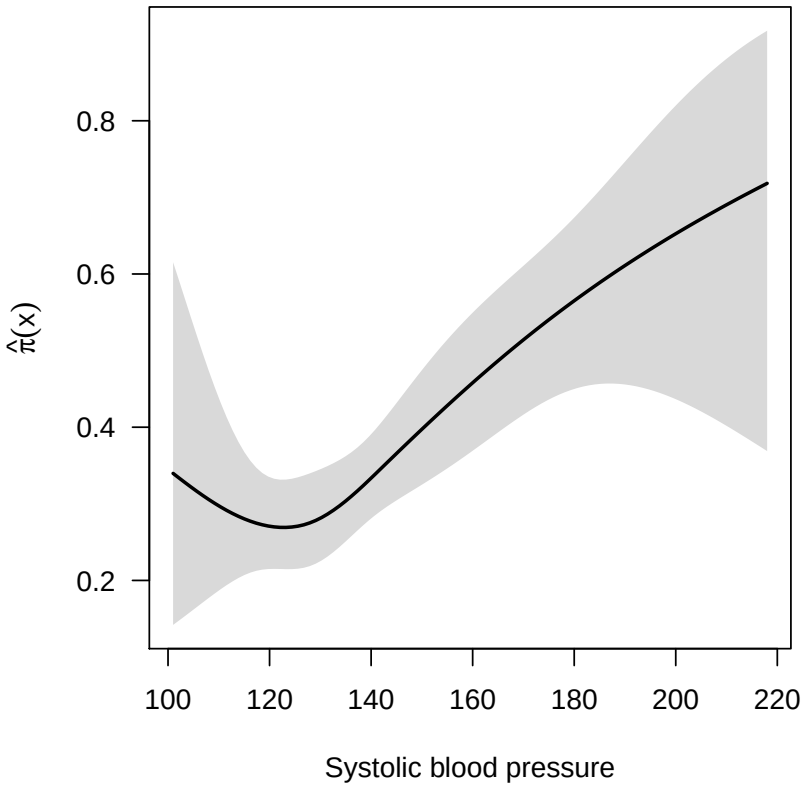
Mean and variance estimation (cont'd)



Mean and variance estimation (cont'd)

Coronary heart disease study, $K = 4$





Problems with knots

- Fixed-df splines are useful tools, but are not truly nonparametric
- Choices regarding the number of knots and where they are located are fundamentally parametric choices and have a large effect on the fit
- Furthermore, assuming that you place knots at quantiles, models will not be nested inside each other, which complicates hypothesis testing
- An alternative, more direct approach is *penalization*

Controlling smoothness with penalization

- Here, we directly solve for the function f that minimizes the following objective function, a penalized version of the least squares objective:

$$\sum_{i=1}^{n}\{y_i - f(x_i)\}^2 + \lambda \int \{f''(u)\}^2 du$$

- The first term captures the fit to the data, while the second penalizes curvature – note that for a line, $f''(u) = 0$ for all u
- Here, λ is the smoothing parameter, and it controls the tradeoff between the two terms:
  - $\lambda = 0$ imposes no restrictions and f will therefore interpolate the data
  - $\lambda = \infty$ renders curvature impossible, thereby returning us to ordinary linear regression

Controlling smoothness with penalization (cont'd)

- This avoids the knot selection problem altogether by formulating the problem in a nonparametric manner
- It may sound impossible to solve for such an f over all possible functions, but the solution turns out to be surprisingly simple: as we will show, the solution to this problem lies in the family of natural cubic splines

Terminology

First, some terminology:

- The parametric splines with fixed degrees of freedom that we have talked about so far are called *regression splines*
- A spline that passes through the points $\{x_i, y_i\}$ is called an *interpolating spline*, and is said to interpolate the points $\{x_i, y_i\}$
- A spline that describes and smooths noisy data by passing close to $\{x_i, y_i\}$ without the requirement of passing through them is called a *smoothing spline*

Natural cubic splines are the smoothest interpolators

Theorem: Out of all twice-differentiable functions passing through the points $\{x_i, y_i\}$, the one that minimizes

$$\lambda \int \{f''(u)\}^2 du$$

is a natural cubic spline with knots at every unique value of $\{x_i\}$

Natural cubic splines solve the nonparametric formulation

Theorem: Out of all twice-differentiable functions, the one that minimizes

$$\sum_{i=1}^{n}\{y_i - f(x_i)\}^2 + \lambda \int \{f''(u)\}^2 du$$

is a natural cubic spline with knots at every unique value of $\{x_i\}$

Design matrix

Let $\{N_j\}_{j=1}^n$ denote the collection of natural cubic spline basis functions and $\mathbf{N}$ denote the $n \times n$ design matrix consisting of the basis functions evaluated at the observed values:

- $\mathbf{N}_{ij} = N_j(x_i)$
- $f(x) = \sum_{j=1}^n N_j(x)\beta_j$
- $f(\mathbf{x}) = \mathbf{N}\boldsymbol{\beta}$

Solution

- **Homework:** Show that the objective function for penalized splines is

$$(\mathbf{y} - \mathbf{N}\boldsymbol{\beta})'(\mathbf{y} - \mathbf{N}\boldsymbol{\beta}) + \lambda\boldsymbol{\beta}'\boldsymbol{\Omega}\boldsymbol{\beta},$$

  where $\boldsymbol{\Omega}_{jk} = \int N_j''(t)N_k''(t)dt$
- The solution is therefore

$$\widehat{\boldsymbol{\beta}} = (\mathbf{N}'\mathbf{N} + \lambda\boldsymbol{\Omega})^{-1}\mathbf{N}'\mathbf{y}$$

Smoothing splines are linear smoothers

- Note that the fitted values can be represented as

$$\begin{aligned}\hat{\mathbf{y}} &= \mathbf{N}(\mathbf{N}'\mathbf{N} + \lambda\boldsymbol{\Omega})^{-1}\mathbf{N}'\mathbf{y} \\ &= \mathbf{L}_\lambda \mathbf{y}\end{aligned}$$

- Thus, smoothing splines are linear smoothers, and we can use all the results that we derived back when discussing local regression

Smoothing splines are linear smoothers (cont'd)

In particular:

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

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

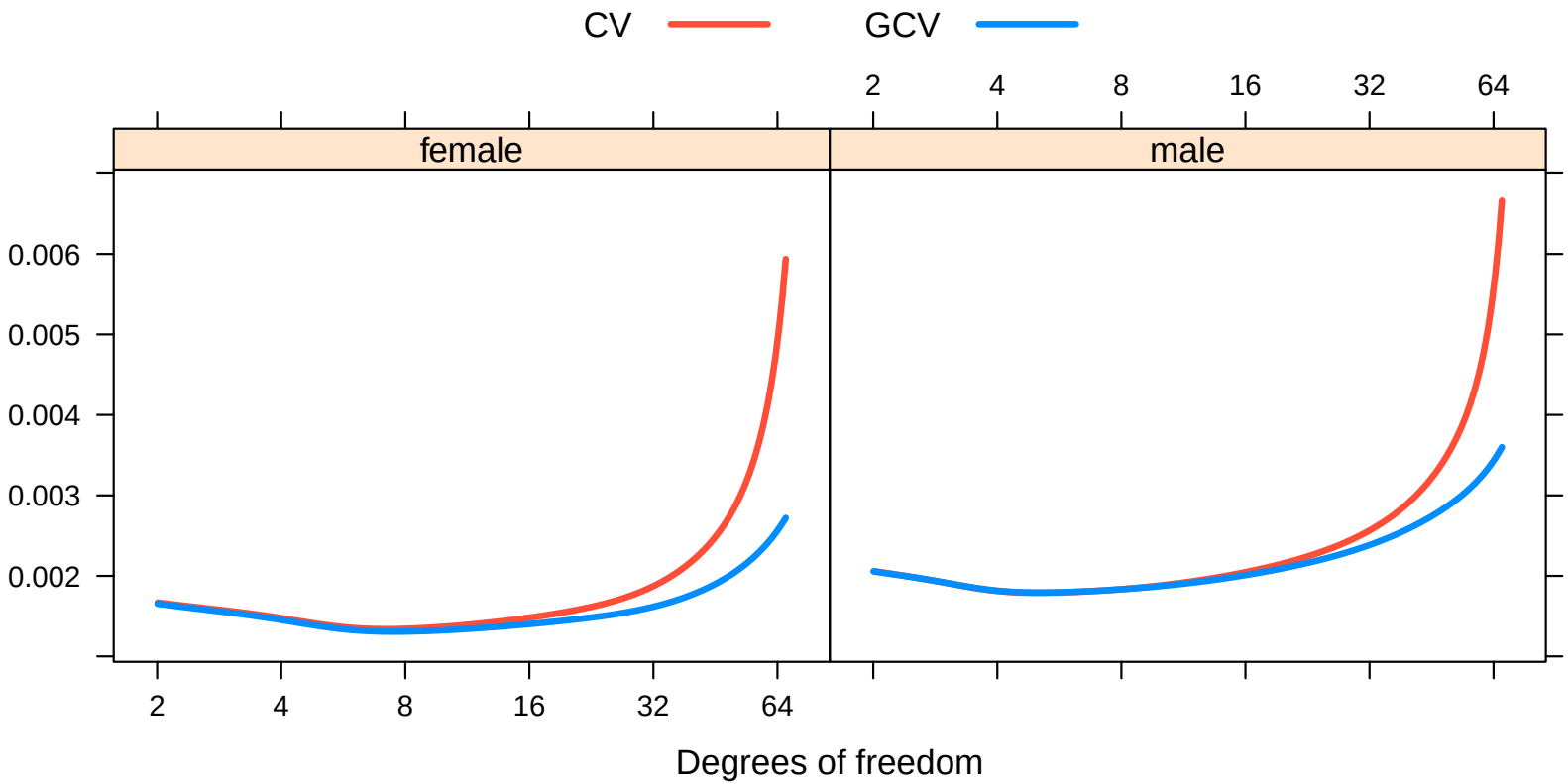
$$\mathbb{V}\hat{f}(x_0) = \sigma^2 \left\| l(x_0) \right\|^2$$

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

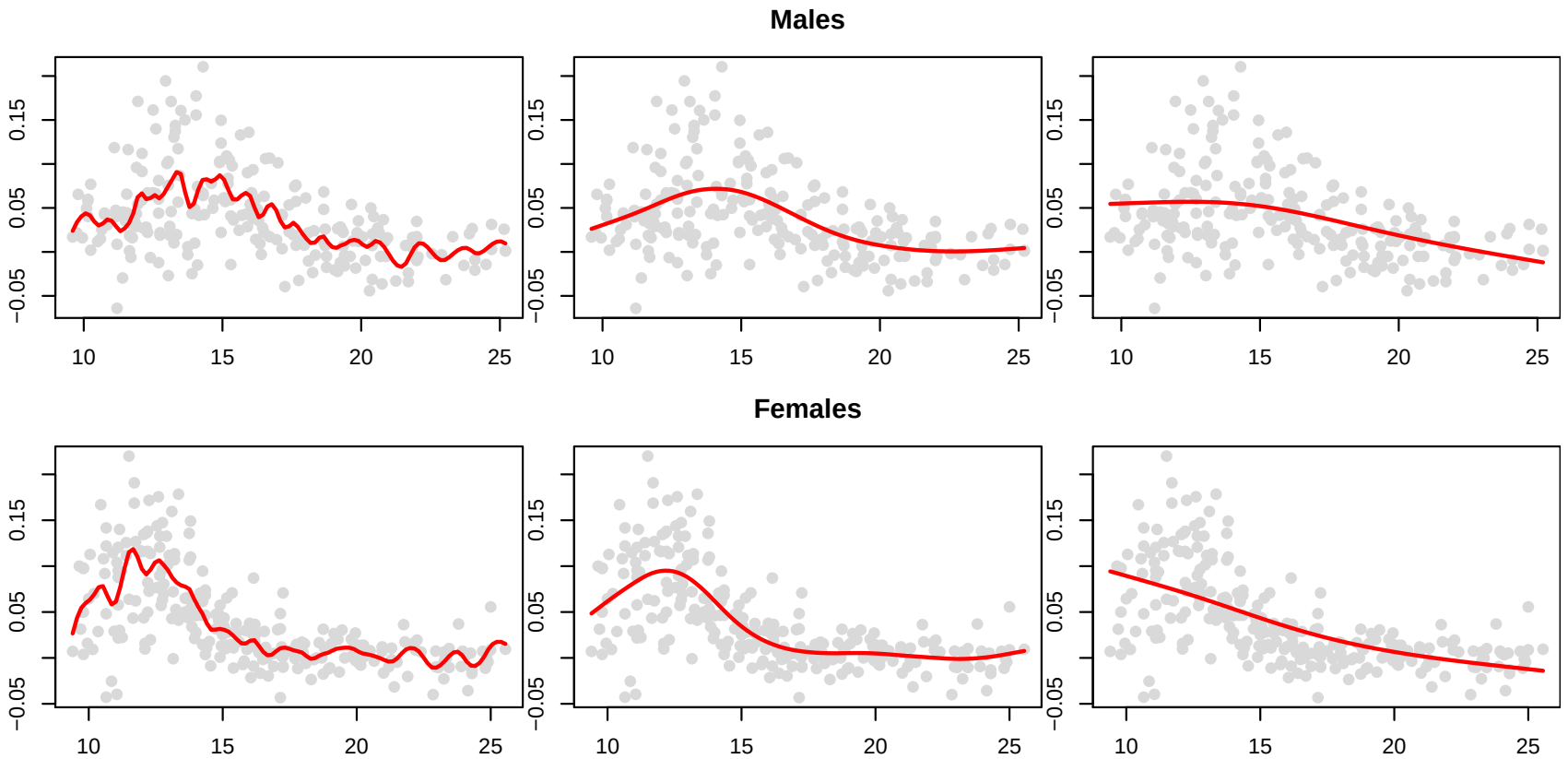
$$\nu = \mathrm{tr}(\mathbf{L})$$

$$\tilde{\nu} = 2\mathrm{tr}(\mathbf{L}) - \mathrm{tr}(\mathbf{L}'\mathbf{L})$$

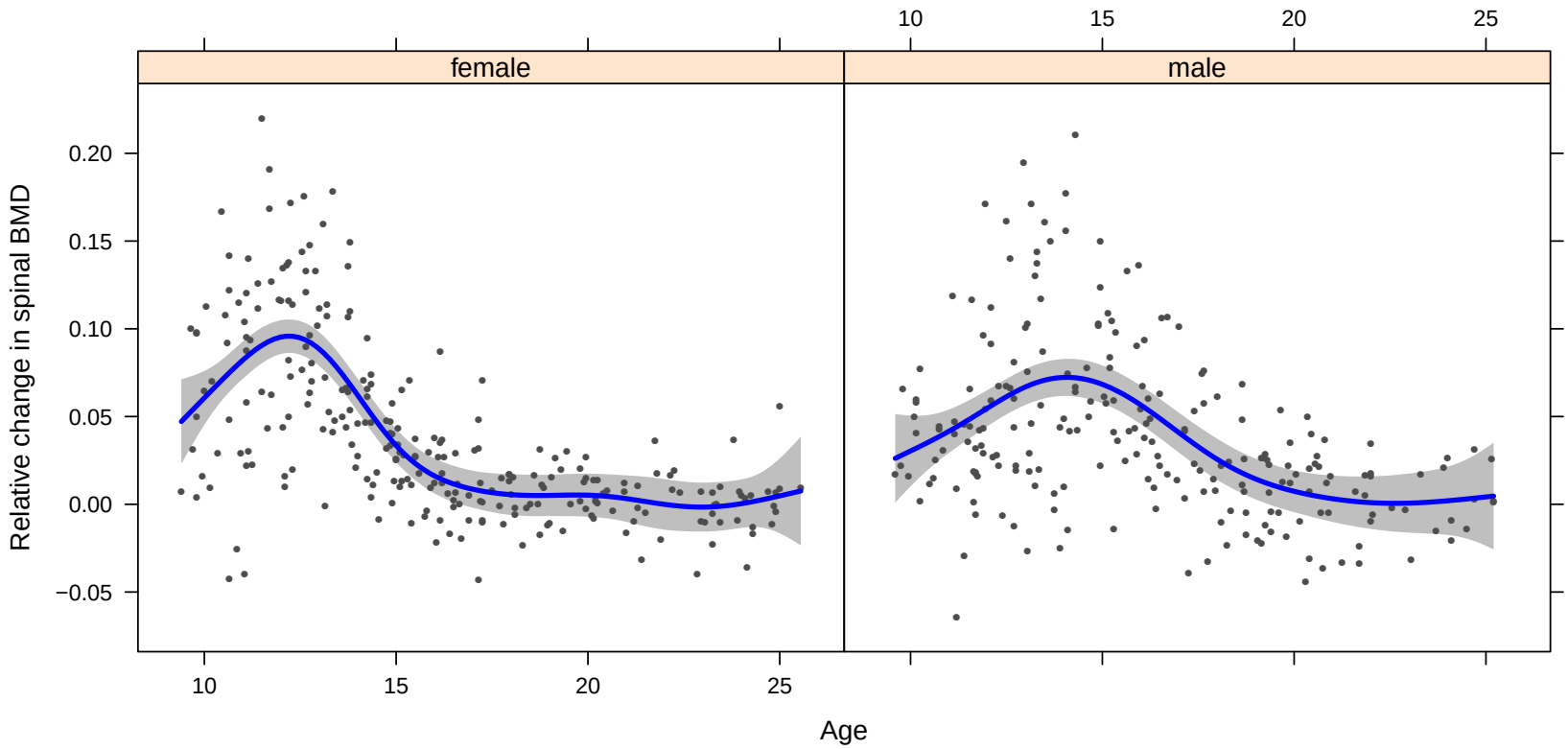
CV, GCV for BMD example



Undersmoothing and oversmoothing of BMD data



Pointwise confidence bands



R implementation

- Recall that local regression had a simple, standard function for basic one-dimensional smoothing (`loess`) and an extensive package for more comprehensive analyses (`locfit`)
- Spline-based smoothing is similar
- `smooth.spline` does not require any packages and implements simple one-dimensional smoothing:

```
fit <- smooth.spline(x,y)
plot(fit,type="l")
predict(fit,xx)
```

- By default, the function will choose λ based on GCV, but this can be changed to CV, or you can specify λ

The mgcv package

- If you have a binary outcome variable or multiple covariates or want confidence intervals, however, `smooth.spline` is lacking
- A very extensive package called `mgcv` provides those features, as well as much more
- The basic function is called `gam`, which stands for *generalized additive model* (we'll discuss GAMs more in a later lecture)
- Note that the main function of the `gam` package was also called `gam`; it is best to avoid having both packages loaded at the same time

The mgcv package (cont'd)

The syntax of gam is very similar to glm and locfit, with a function s() placed around any terms that you want a smooth function of:

```
fit <- gam(y~s(x))
fit <- gam(y~s(x),family="binomial")
plot(fit)
plot(fit,shade=TRUE)
predict(fit,newdata=data.frame(x=xx),se.fit=TRUE)
summary(fit)
```

Hypothesis testing

- As with local regression, we are often interested in testing whether the smaller of two nested models provides an adequate fit to the data
- As before, we may construct F tests and generalized likelihood ratio tests:

$$\begin{aligned} F &= \frac{(\text{RSS}_0 - \text{RSS}_1)/q}{\hat{\sigma}^2} \\ &\dot{\sim} F_{q,n-\nu_1} \\ \Lambda &= 2\left\{l(\widehat{\boldsymbol{\beta}}|\mathbf{y}) - l(\widehat{\boldsymbol{\beta}}_0|\mathbf{y})\right\} \\ &\dot{\sim} \chi^2_q \end{aligned}$$

where q is the difference in degrees of freedom between the two models

Hypothesis testing (cont'd)

- As with local regression, these tests do not strictly preserve type I error rates, but still provide a way to compute useful approximate p-values in practice
- Like the `gam` package, `mgcv` provides an `anova` method:

```
anova(fit0,fit,test="F")
anova(fit0,fit,test="Chisq")
```

- It should be noted, however, that such tests treat λ as fixed, even though in reality it is often estimated from the data using CV or GCV