MONASH University

MONASH BUSINESS SCHOOL

ETC3580: Advanced Statistical Modelling

Week 10: Nonparametric inference

Outline

General kernel form of linear smoothers

Linear smoother

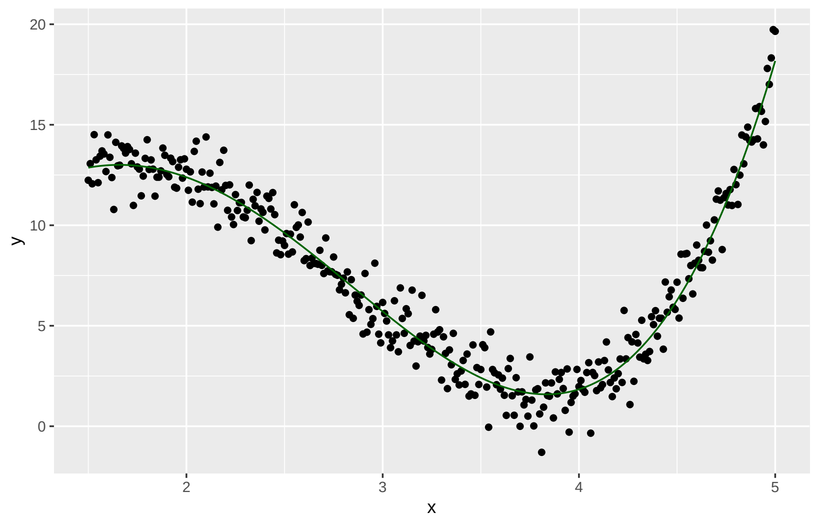
$$\hat{f}(x) = \sum_{j=1}^{n} w_j(x) y_j$$

- Nadaraya-Watson smoothing:

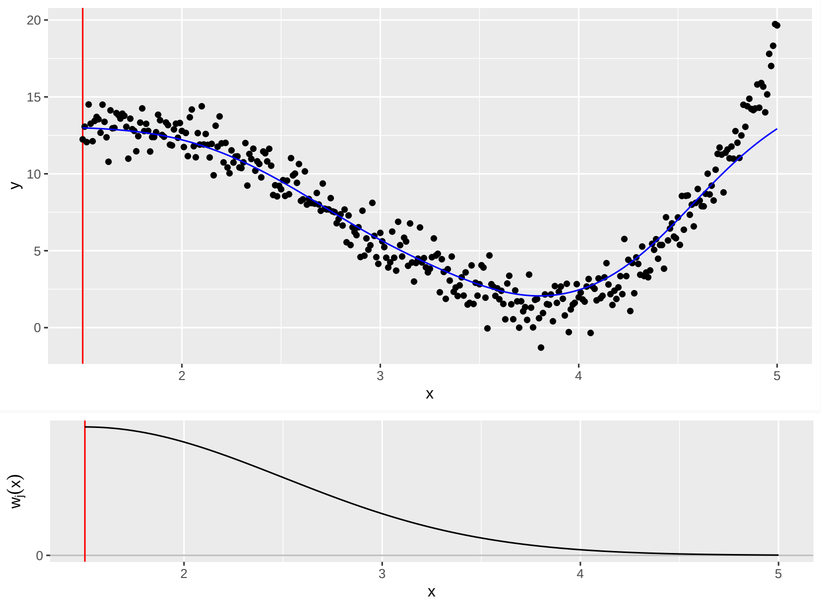
$$w_j(x) = \frac{K\left(\frac{x - x_j}{h}\right)}{\sum_{i=1}^{n} K\left(\frac{x - x_i}{h}\right)}$$

- Almost all smoothing methods can be written in this form for different functions $w_j(x)$

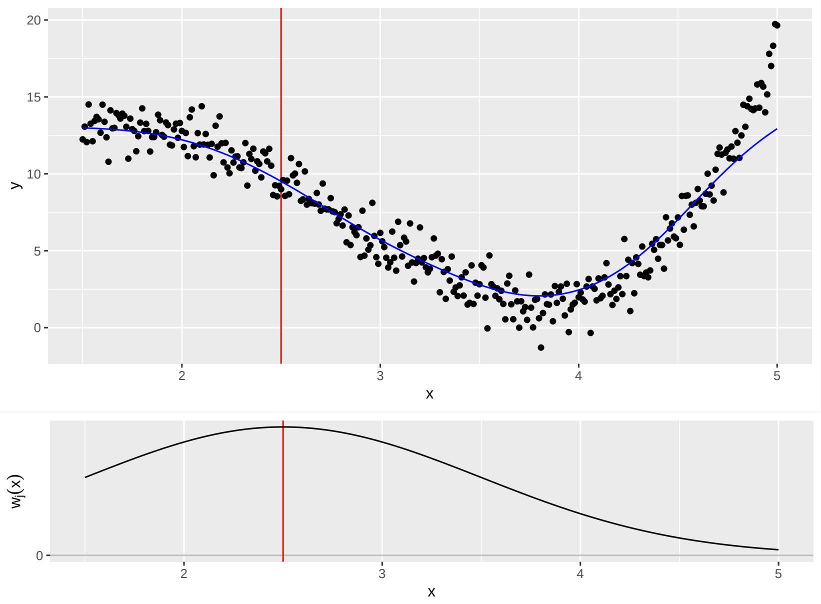
Example



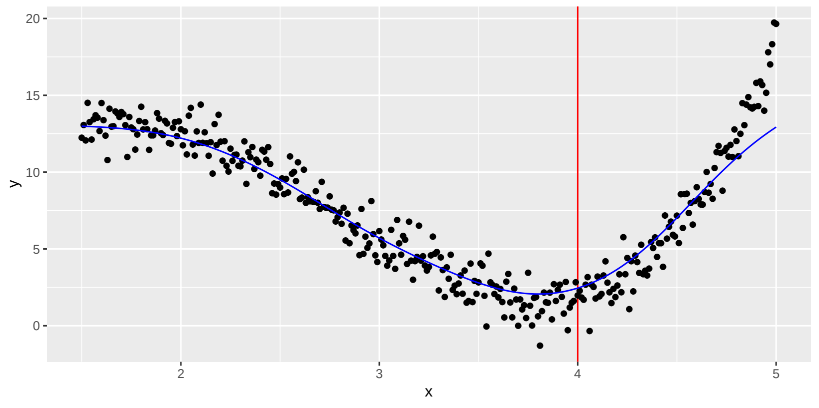
Kernel estimator

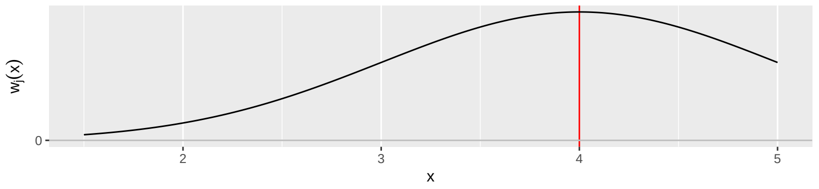


Kernel estimator

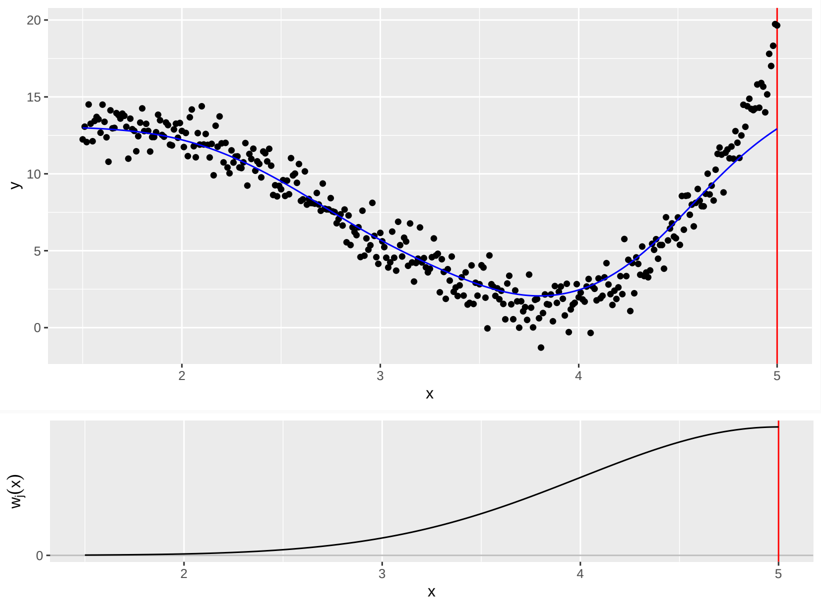


Kernel estimator





Kernel estimator



Local polynomial estimator

Assume

$$f(u) = a_0 + a_1(u - x) + \cdots + a_p(u - x)^p.$$

Then the coefficients, $\hat{a}_i$, are the values of a_i which minimise

$$\text{WLS}(x) = \sum_{j=1}^{n} w_j(x)\Big(y_j - a_0 - a_1(x_j - x) - \cdots - a_p(x_j - x)^p\Big)^2$$

and $\hat{f}(x) = \hat{a}_0$. In matrix notation we can write

$$\text{WLS}(x) = (\boldsymbol{Y} - X\boldsymbol{a})'W(x)(\boldsymbol{Y} - X\boldsymbol{a})$$

where $[X]_{ji} = (x_j - x)^i$ and $W(x)$ is the diagonal matrix with elements $w_j(x)$.

Local polynomial estimator

The minimizer of this function is

$$\hat{\boldsymbol{a}} = (X'W(x)X)^{-1}X'W(x)\boldsymbol{Y}.$$

Therefore,

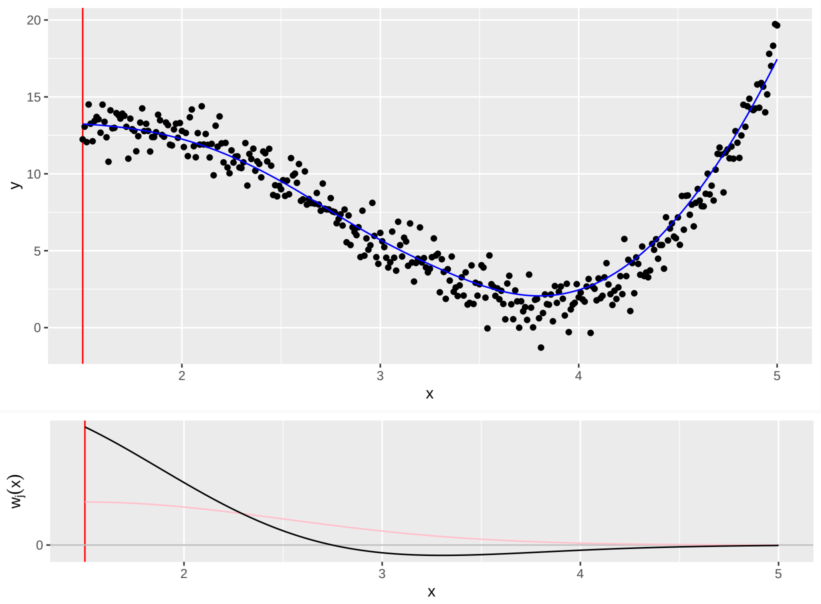
$$\hat{f}(x) = [1, 0, \ldots, 0](X'W(x)X)^{-1}X'W(x)\boldsymbol{Y} = \sum_{j=1}^{n} l_j(x)y_j$$

where

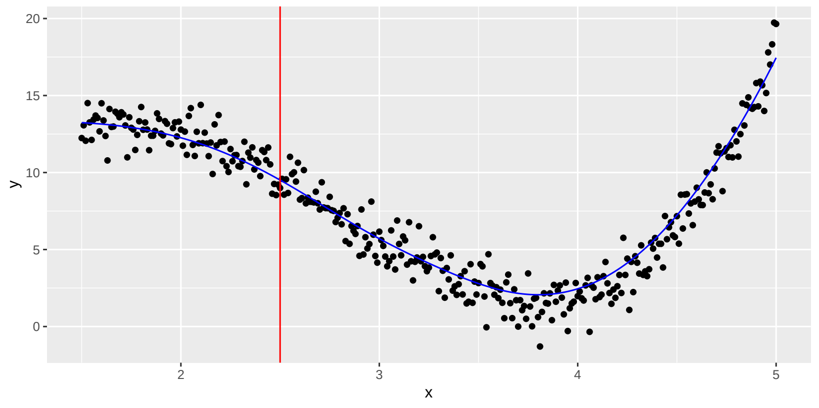
$$l_j(x) = [1, 0, \ldots, 0](X'W(x)X)^{-1}[1, (x_j - x), \ldots, (x_j - x)^p]w_j(x)$$

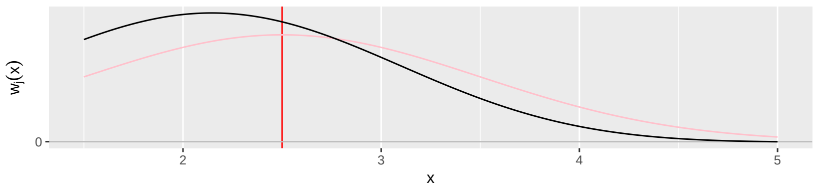
So a local polynomial is equivalent to a kernel smoother but with an unusual weight function. We call the weights $l_j(x)$ the *effective kernel* at x. If $p = 0$, then $l_j(x) = w_j(x)$.

Local linear estimator

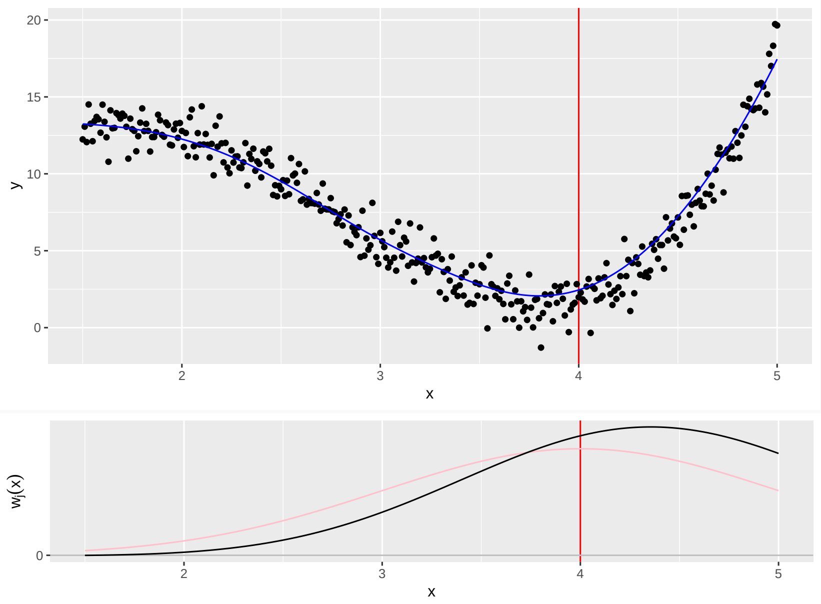


Local linear estimator

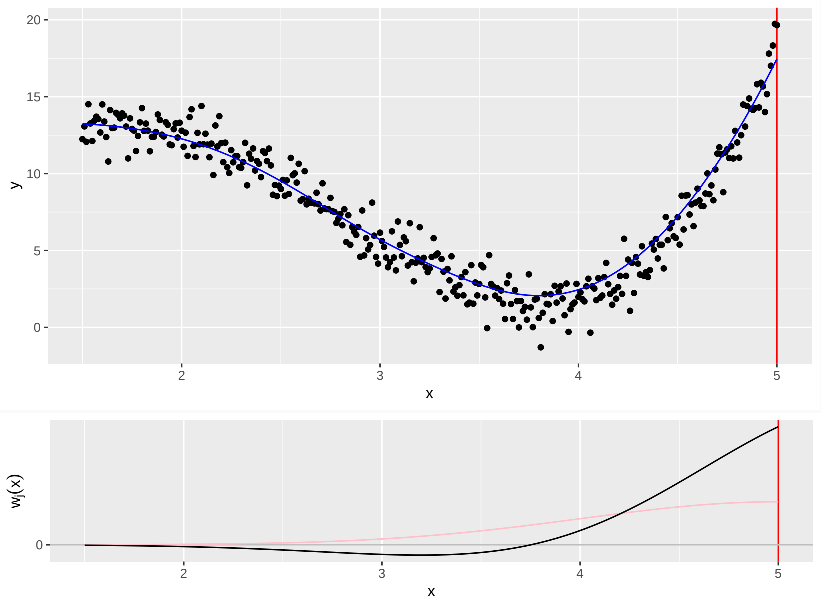




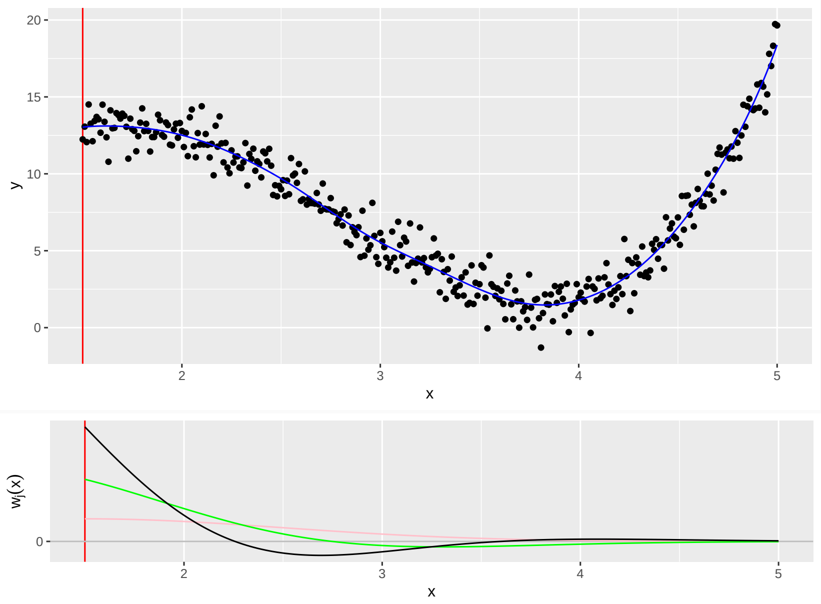
Local linear estimator



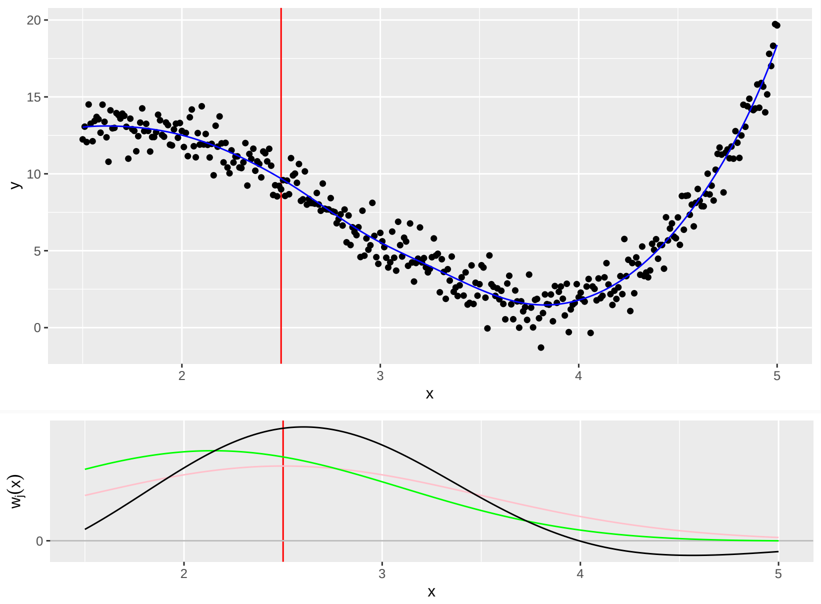
Local linear estimator



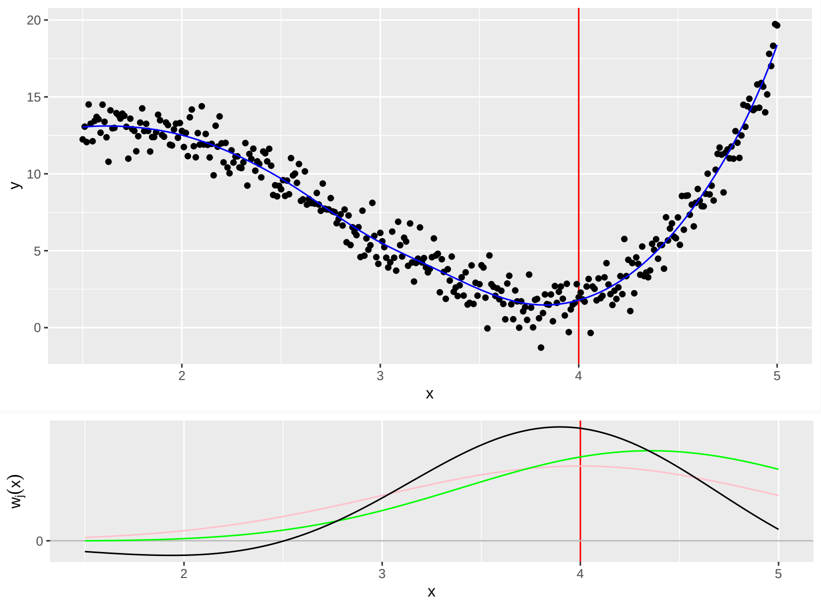
Local quadratic smoother



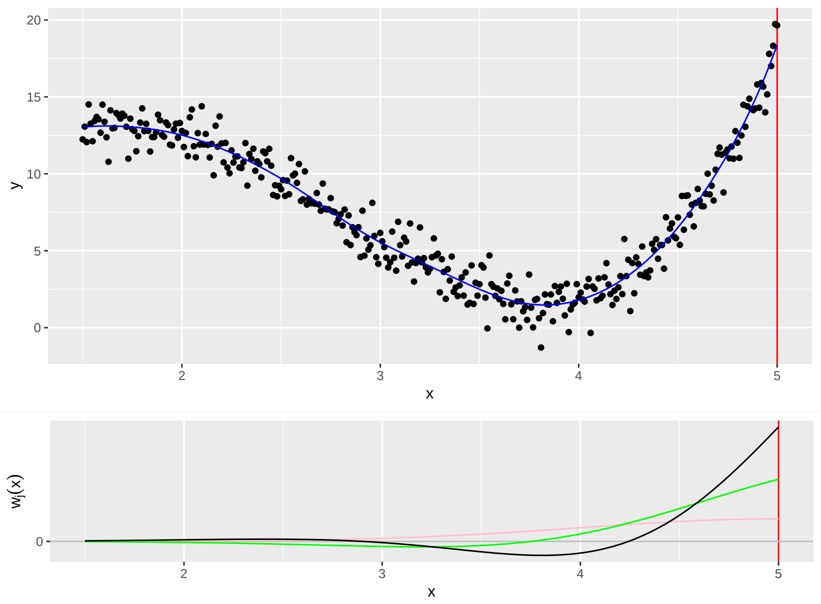
Local quadratic smoother



Local quadratic smoother



Local quadratic smoother

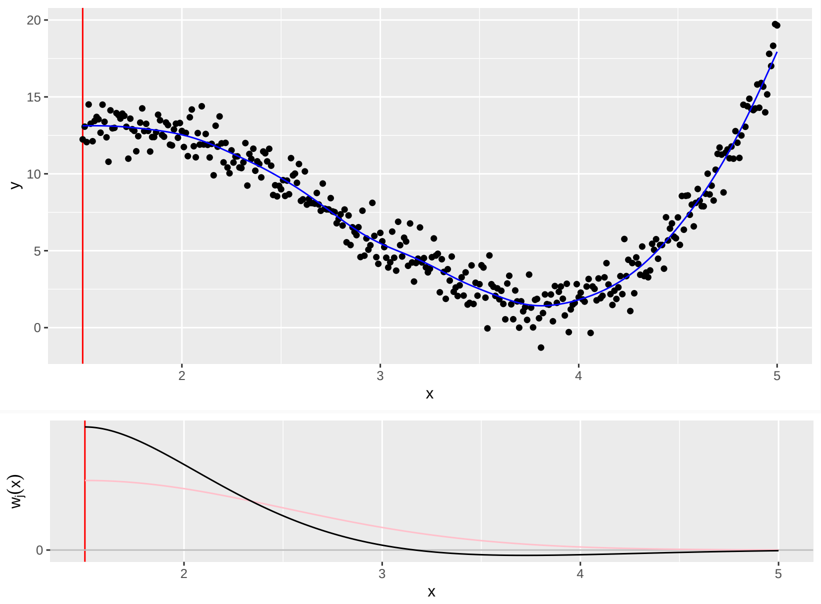


Smoothing spline

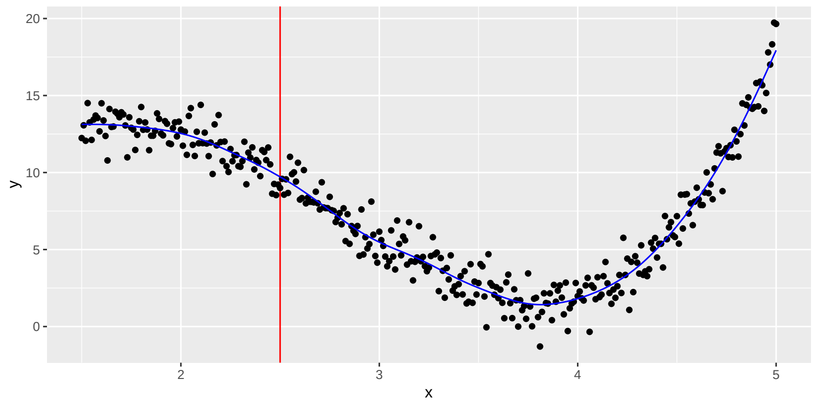
- A cubic smoothing spline can also be written as a kernel smoother with kernel function asymptotically equal to

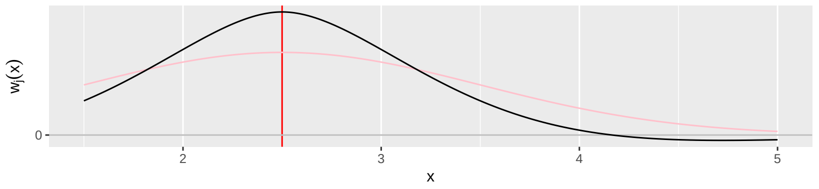
$$K_s(u) = \frac{1}{2} \exp\left(-\frac{|u|}{h\sqrt{2}}\right) \sin\left(\frac{|u|}{h\sqrt{2}} + \frac{\pi}{4}\right).$$

Smoothing spline

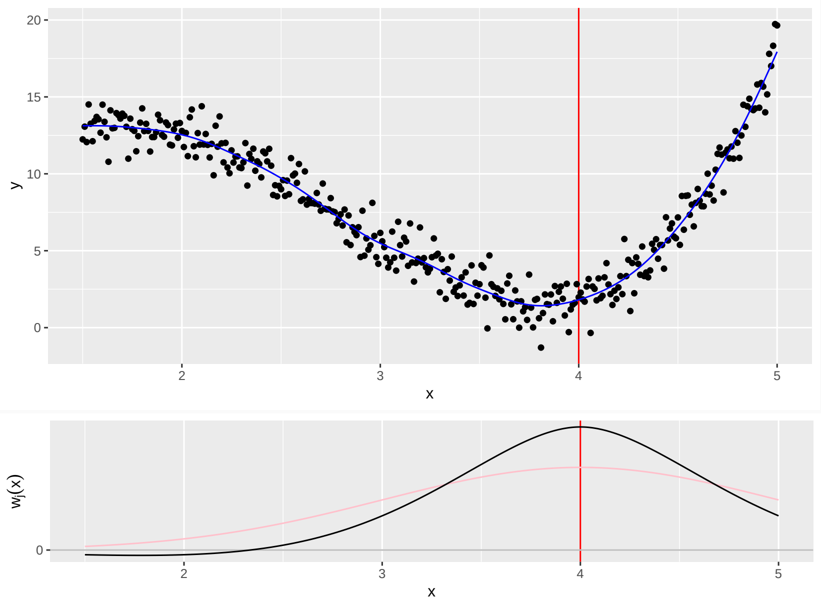


Smoothing spline

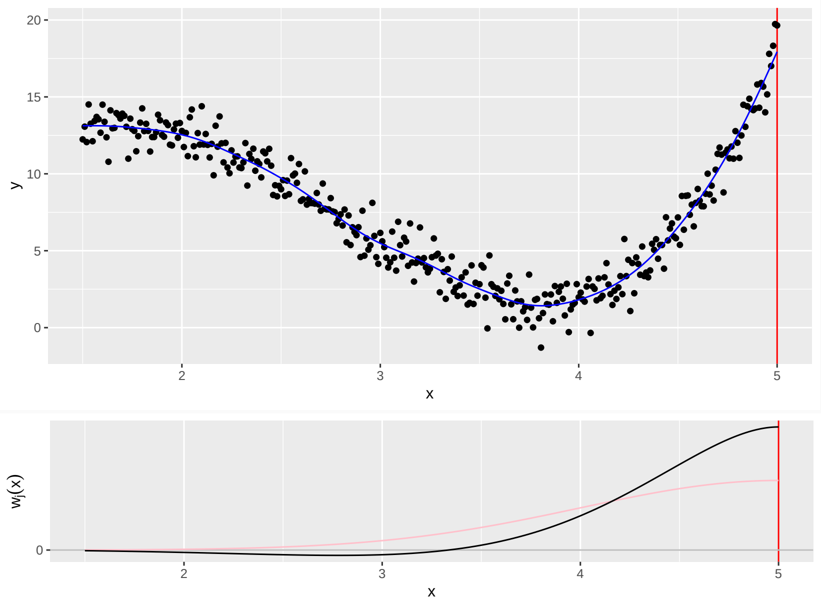




Smoothing spline



Smoothing spline



Regression splines

Regression splines are linear models, and so fitted values can be written as

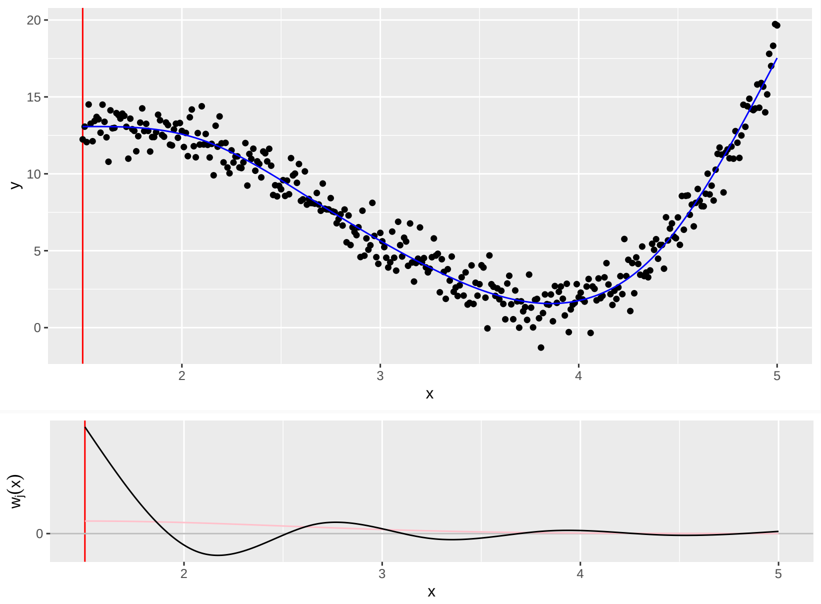
Therefore,

$$\hat{f}(x) = \boldsymbol{x}^{*\prime}(X'X)^{-1}X'\boldsymbol{Y} = \sum_{j=1}^{n} l_j(x)y_j$$

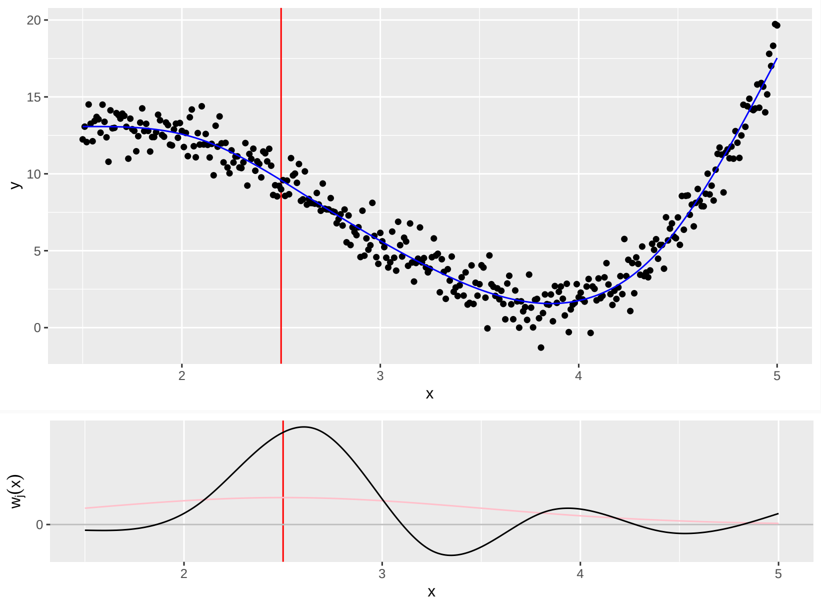
where

$$l_j(x) = \boldsymbol{x}^{*\prime}(X'X)^{-1}\boldsymbol{x}^{*\prime}$$

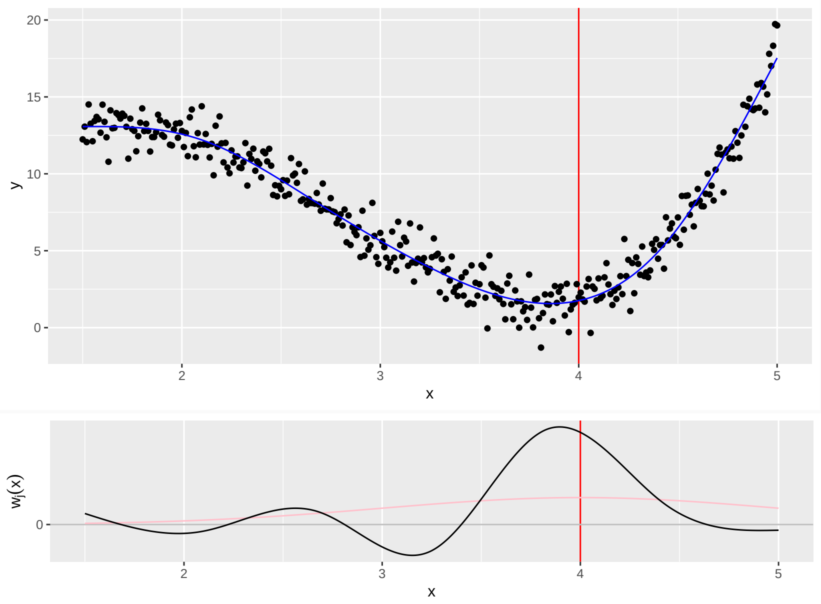
Regression splines



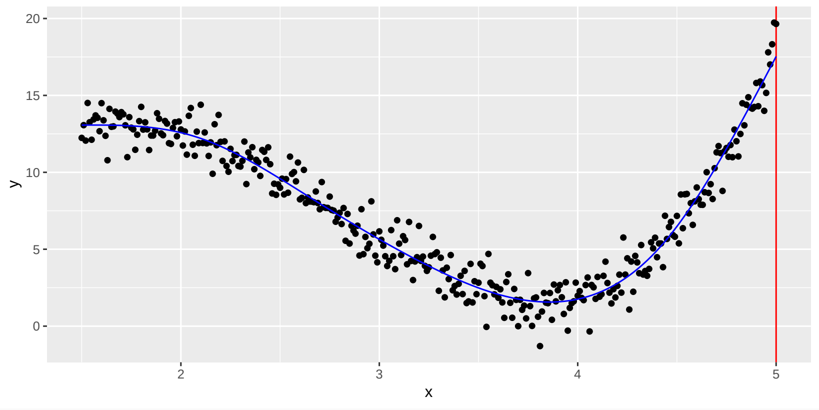
Regression splines

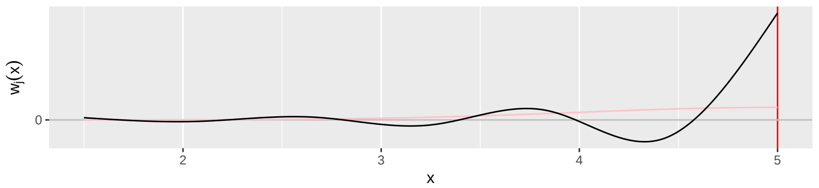


Regression splines



Regression splines





Outline

Inference for linear smoothers

All of the methods we have looked at can be written in the form

$$\hat{f}(x) = \sum_{j=1}^{n} w_j(x) y_j.$$

Thus they are linear in the observations. The set of weights, $w_j(x)$, is known as the equivalent kernel at x. Let $\hat{\boldsymbol{f}} = [\hat{f}(x_1), \hat{f}(x_2), \dots, \hat{f}(x_n)]'$. Then

$$\hat{\boldsymbol{f}} = \boldsymbol{S}\boldsymbol{y}$$

where $\boldsymbol{S} = [w_j(x_i)]$ is an $n \times n$ matrix that we call a *smoother matrix*.

Inference for linear smoothers

- The rows of $\boldsymbol{S}$ are the equivalent kernels for producing fits at each of the observed values $x_1, \ldots, x_n$.
- Any reasonable smoother should preserve a constant function so that $\boldsymbol{S}\mathbf{1} = \mathbf{1}$ where $\mathbf{1}$ is a vector of ones. This implies that the sum of the weights in each row is one.
- The matrix $\boldsymbol{S}$ is analogous to the hat matrix $\boldsymbol{X}(\boldsymbol{X}'\boldsymbol{X})^{-1}\boldsymbol{X}'$ in a standard linear model.

Degrees of freedom

Want: Approximate df for our linear smoothers.

- high df for very wiggly smoothers
- low df for very smooth smoothers.
- Least squares regression: $\boldsymbol{S} = \boldsymbol{X}(\boldsymbol{X}'\boldsymbol{X})^{-1}\boldsymbol{X}'$

$$
\begin{aligned}
\gamma = \text{df} &= \text{\# linearly independent predictors in model} \\
&= \text{rank}(\boldsymbol{S}) \\
&= \text{tr}(\boldsymbol{S}) \\
&= \text{tr}(\boldsymbol{S}\boldsymbol{S}') \\
&= \text{tr}(2\boldsymbol{S} - \boldsymbol{S}\boldsymbol{S}').
\end{aligned}
$$

Any of these could be used for df of general linear smoother.

Estimating the variance

- Linear regression: error has $n - \gamma$ df.
- Hence define df of error for a linear smoother as $n - \gamma$ where $\gamma = \mathrm{tr}(\boldsymbol{S})$.
- Assuming zero bias for smoother, an unbiased estimator of σ^2 is given by

$$\hat{\sigma}^2 = \frac{1}{n - \gamma} \sum_{j=1}^{n} (y_j - \hat{f}(x_j))^2.$$

Confidence intervals

$\text{Cov}(\hat{\boldsymbol{f}}) = \boldsymbol{S}\boldsymbol{S}'\sigma^2$

Assuming negligible bias, approximate 95% CI for $\boldsymbol{f}$ are:

$$\hat{\boldsymbol{f}} \pm 1.96\hat{\sigma}\sqrt{\text{diag}(\boldsymbol{S}\boldsymbol{S}')}.$$

- Pointwise intervals. (i.e,. 95% CI for each value of x.)
- On average, true value of $f(x)$ lies outside these intervals 5% of the time.

Approximate F tests

Approximate F tests using the approximate df.

To compare two smooths:

$$\hat{\boldsymbol{f}}_1 = \boldsymbol{S}_1\boldsymbol{y} \qquad (\text{df} = \gamma_1)$$

$$\hat{\boldsymbol{f}}_2 = \boldsymbol{S}_2\boldsymbol{y} \qquad (\text{df} = \gamma_2).$$

$\gamma_i = \text{df} = \text{tr}(2\boldsymbol{S}_i - \boldsymbol{S}_i\boldsymbol{S}_i')$ for each of the models $i = 1, 2$.

Let RSS_1 and RSS_2 be residual sum of squares for each smoother.

$$\frac{(\text{RSS}_1 - \text{RSS}_2)/(\gamma_2 - \gamma_1)}{\text{RSS}_2/(n - \gamma_2)} \sim F_{\gamma_2-\gamma_1, n-\gamma_2}.$$

- Implemented by `anova` in R

Applications

Test for linearity

Let $\hat{f}_1$ represent a linear regression and we wish to test if the linearity is real by fitting a nonparametric nonlinear smooth curve $\hat{f}_2$.

Applications

Test for linearity

Let $\hat{f}_1$ represent a linear regression and we wish to test if the linearity is real by fitting a nonparametric nonlinear smooth curve $\hat{f}_2$.

Test for bias in residuals

After fitting a model, the residuals can be modelled as a function of the predictor variable. If the function is not significantly different from the zero function, there is no significant bias.

Bias and variance

The bias vector is $\boldsymbol{b} = \boldsymbol{f} - \mathrm{E}(\boldsymbol{S}\boldsymbol{y}) = \boldsymbol{f} - \boldsymbol{S}\boldsymbol{f} = (\boldsymbol{I} - \boldsymbol{S})\boldsymbol{f}$.

Then we can compute the mean square error as

$$\begin{aligned} \mathrm{MSE} &= \frac{1}{n}\sum_{j=1}^{n} \mathrm{Var}(\hat{f}_i) + \frac{1}{n}\sum_{j=1}^{n} b_i^2 \\ &= \frac{\mathrm{tr}(\boldsymbol{S}\boldsymbol{S}')}{n}\sigma^2 + \frac{\boldsymbol{b}'\boldsymbol{b}}{n} \end{aligned}$$

The first term measures variance while the second measures squared bias.

- Smoothing is a bias-variance tradeoff

Cross-validation

Find h which minimises cross-validation function

$$\text{CV}(h) = \frac{1}{n}\sum_{j=1}^{n}[\hat{f}_j(x_j) - y_j]^2$$

$$\hat{f}_j(x) = \frac{1}{1 - w_j(x)} \sum_{\substack{i=1 \\ i \neq j}}^{n} w_i(x) y_i.$$

- residuals: $\hat{e}_j = y_j - \hat{f}(x_j)$
- LOO residuals: $\hat{e}_{(j)} = y_j - \hat{f}_j(x_j)$

Use same computational trick as for LM to avoid computing n separate smoothers.

Cross-validation

$$
\begin{aligned}
\hat{e}_{(j)} &= y_j - \hat{f}_j(x_j) \\
&= y_j - \frac{1}{1 - w_j(x_j)} \sum_{\substack{i=1 \\ i \neq j}}^{n} w_i(x_j) y_i . \\
&= y_j - \frac{1}{1 - w_j(x_j)} \left(\hat{f}(x_j) - w_j(x_j) y_j \right) \\
&= y_j \left(1 + \frac{w_j(x_j)}{1 - w_j(x_j)} \right) - \frac{1}{1 - w_j(x_j)} \hat{f}(x_j) \\
&= y_j \left(\frac{1 - w_j(x_j) + w_j(x_j)}{1 - w_j(x_j)} \right) - \frac{1}{1 - w_j(x_j)} \hat{f}(x_j) \\
&= \left(y_j - \hat{f}(x_j) \right) \frac{1}{1 - w_j(x_j)} = \frac{\hat{e}_j}{1 - w_j(x_j)}
\end{aligned}
$$

Cross-validation

$$\mathrm{CV}(h) = \frac{1}{n}\sum_{j=1}^{n} \hat{e}_{(j)}^2 = \frac{1}{n}\sum_{j=1}^{n} \hat{e}_j^2 \left(1 - w_j(x_j)\right)^{-2}$$

Cross-validation

$$\mathrm{CV}(h) = \frac{1}{n}\sum_{j=1}^{n} \hat{e}_{(j)}^2 = \frac{1}{n}\sum_{j=1}^{n} \hat{e}_j^2 \left(1 - w_j(x_j)\right)^{-2}$$

CV(h) is a **penalized mean squared error**.

Cross-validation

$$\text{CV}(h) = \frac{1}{n}\sum_{j=1}^{n} \hat{e}_{(j)}^2 = \frac{1}{n}\sum_{j=1}^{n} \hat{e}_j^2 \left(1 - w_j(x_j)\right)^{-2}$$

CV(h) is a **penalized mean squared error**.

Generalization:

Find h which minimises penalized MSE:

$$G(h) = \frac{1}{n}\sum_{j=1}^{n}[\hat{f}(x_j) - y_j]^2 p(w_j(x_j))$$

where $p(u)$ is a penalty function.

CV: $p(u) = (1-u)^{-2}$.

Penalized MSE

Examples:

Shibata's selector	$p(u) = 1 + 2u$
Generalized cross-validation	$p(u) = (1 - u)^{-2}$
Akaike's information criterion	$p(u) = \exp(2u)$
Finite prediction error	$p(u) = (1 + u)/(1 - u)$
Rice's T	$p(u) = (1 - 2u)^{-1}$

- Goal is to penalize small bandwidths.
- $w_j(x_j) \to 1$ as $h \to 0$ and $w_j(x_j) \to 0$ as $h \to \infty$.
- Different $p(u)$ almost equal for large h but penalize small h differently.

Penalized MSE

- `mgcv::gam` function uses GCV.
- If $\hat{h}$ is minimising bandwidth of $G(h)$ and $\hat{h}_0$ is MSE optimal bandwidth, then

$$\frac{\text{MSE}(\hat{h})}{\text{MSE}(\hat{h}_0)} \xrightarrow{p} 1 \qquad \text{and} \qquad \frac{\hat{h}}{\hat{h}_0} \xrightarrow{p} 1.$$

Outline

Derivative estimation

$$\hat{f}(x) = \sum_{j=1}^{n} w_j(x) y_j \qquad \Rightarrow \qquad \hat{f}^{(k)}(x) = \sum_{j=1}^{n} w_j^{(k)}(x) y_j.$$

- if $w_j(x)$ is not smooth, then $w_j^{(k)}(x)$ will have some discontinuities.
- To obtain smooth estimate of $\hat{f}^{(k)}(x)$, we need $w_j(x)$ to have continuous derivatives up to order k. This rules out many of the standard kernel weighting functions.

Derivative estimation

For an asymptotically unbiased estimator of $f'(x)$, we require
$$\sum_{j=1}^{n} w_j^{(1)}(x) = 0$$

and
$$\sum_{j=1}^{n} w_j^{(1)}(x)(x - x_j) = 1.$$

- Local polynomials of degree $p \geq 1$ will satisfy these constraints.
- So will cubic splines (of any flavour)
- But not kernel smooths.

Derivative estimation

For an asymptotically unbiased estimator of $f''(x)$, we require

$$\sum_{j=1}^{n} w_j^{(2)}(x) = 0$$

$$\sum_{j=1}^{n} w_j^{(2)}(x)(x - x_j) = 0$$

and $$\sum_{j=1}^{n} w_j^{(2)}(x)(x - x_j)^2 = 2.$$

- Local polynomials of degree $p \geq 2$ will satisfy these constraints.
- So will cubic splines (of any flavour)
- But not kernel or locally linear smoothers.

Outline

Multidimensional kernel smoothing

If $m \geq 2$ predictors, need to fit surface rather than line.

Multidimensional kernel smoothing

$$\hat{f}(\mathbf{z}) = \sum_{j=1}^{n} w_j(\mathbf{z}) y_j \quad \text{where} \quad w_j(\mathbf{z}) = \frac{K_m(\mathbf{z} - \mathbf{x}_j)}{\sum_{j=1}^{n} K_m(\mathbf{z} - \mathbf{x}_j)}.$$

$\mathbf{z}$ and $\mathbf{x}_j$ are m-dimensional vectors and $K_m(\mathbf{u})$ is an m-dimensional function.

Product kernel: $K_m(\mathbf{u}) = \prod_{i=1}^{m} \frac{1}{h_i} K(u_i/h_i)$ where $K(u)$ is univariate kernel and h_i is smoothing parameter in ith dimension.

Multidimensional distance: $K_m(\mathbf{u}) = \frac{1}{h} K(\|u\|/h)$ where $\|u\|$ is distance metric (e.g. Euclidean distance). Only one smoothing parameter, h, used.

Multidimensional kernel smoothing

- If multidimensional distance used, it is usually necessary to standardise each predictor by dividing by its standard deviation or some other measure of spread.
- If $m = 1$, both methods give the standard univariate results.

Local polynomial surfaces

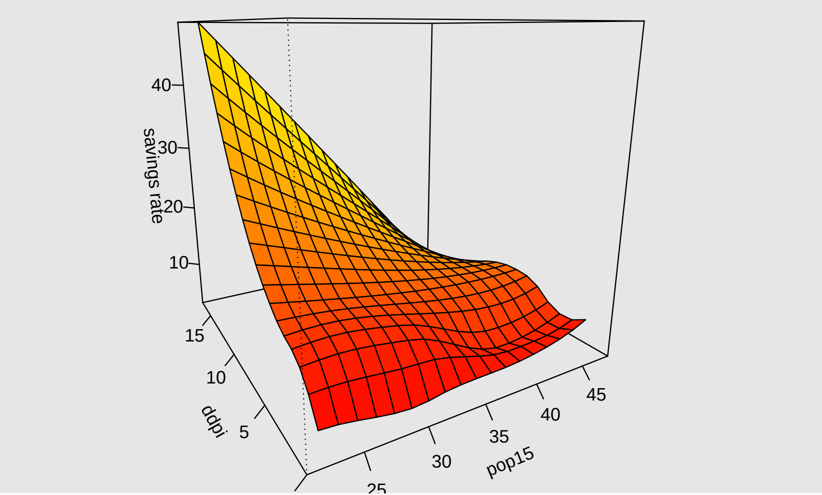
Locally weighted lines generalise easily to higher dimensions.

- Instead of computing local lines, compute a local plane.
- If predictors are *w* and *v*, local plane is computed using multiple regression on *w* and *v*.
- Local quadratic surfaces computed using multiple regression on w, v, wv, w^2 and v^2.

```
fit <- loess(y ~ x + z, span)
```

Bivariate smoothing

```
lomod <- loess(sr ~ pop15 + ddpi, data=savings)
```



Bivariate splines

Smoothing splines can be generalized to thin-plate splines in two dimensions.

- Minimize

$$\sum_{i=1}^{n}(y_i - f(\mathbf{x}_i))^2 + \lambda \iint \left[\left(\frac{\partial^2 f}{\partial x_1^2}\right) + 2\left(\frac{\partial^2 f}{\partial x_1 \partial x_2}\right) + \left(\frac{\partial^2 f}{\partial x_2^2}\right)\right] dx_1\, dx_2.$$

- In R:

```r
library(mgcv)
fit <- gam(y ~ s(x, z), data)
vis.gam(fit)
```

Bivariate smoothing

```r
library(mgcv)
smod <- gam(sr ~ s(pop15, ddpi), data=savings)
```