Nonlinear Modeling I

Splines and Polynomial Regression (close to chapter 7, ISLR)

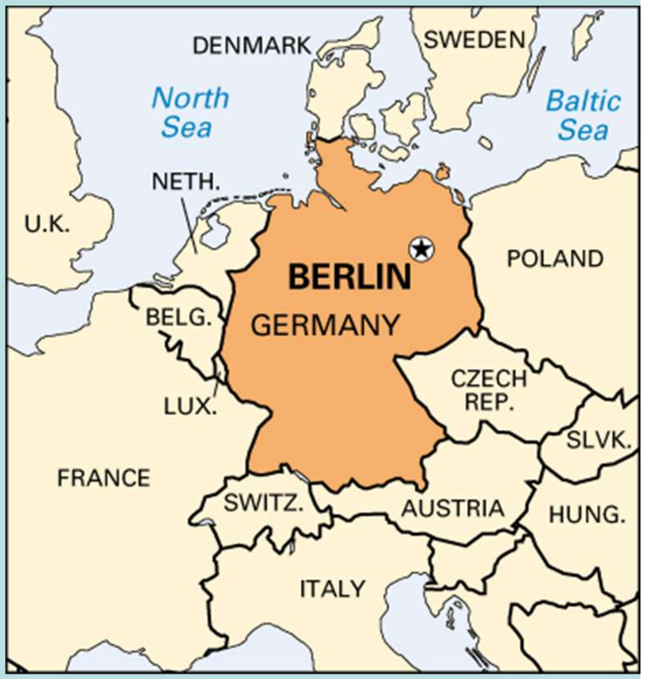
M Loecher

Contents

Nonlinear Modeling in R

Expanding the R user‘s toolbox with splines, GAMs, trees and forests

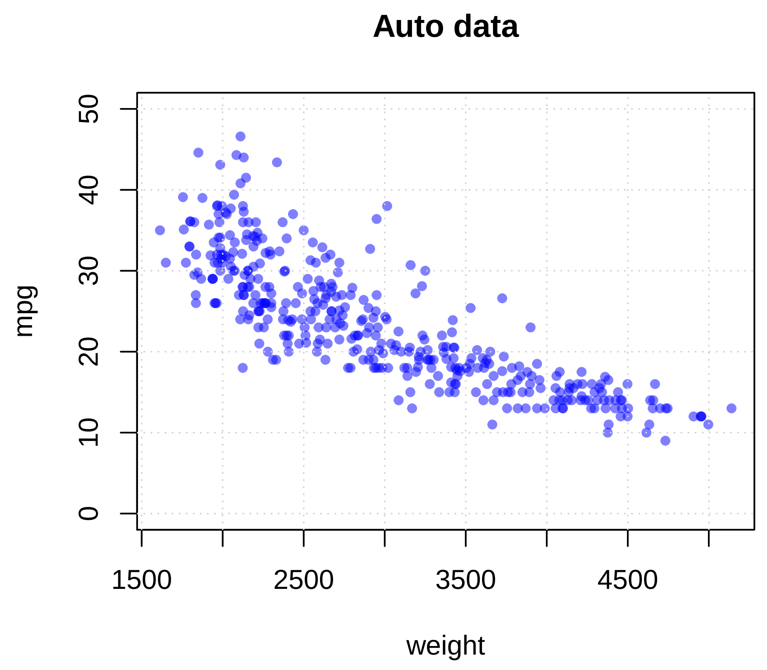
Markus Löcher
Berlin School of Economics and Law

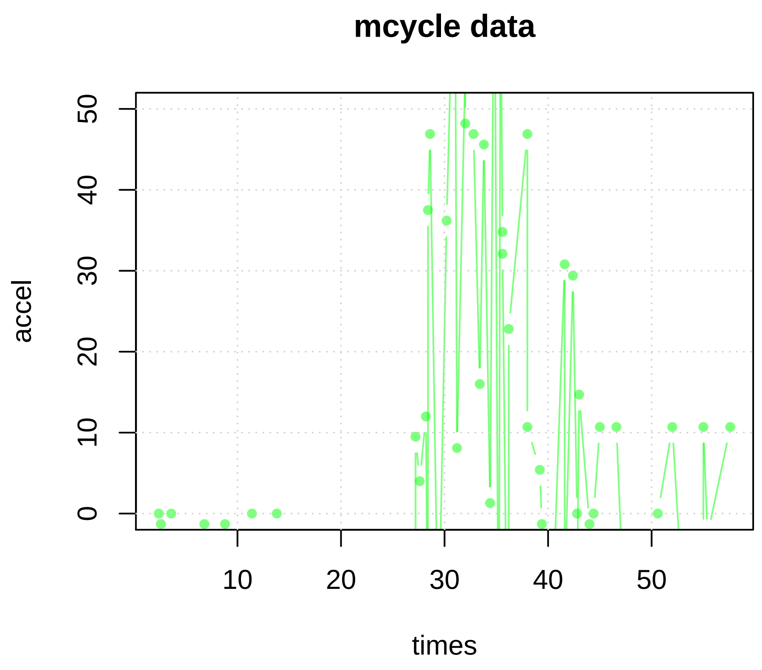

DENMARK
SWEDEN
North Sea
Baltic Sea
NETH.
U.K.
BERLIN
GERMANY
POLAND
BELG.
CZECH REP.
LUX.
SLVK.
FRANCE
SWITZ.
AUSTRIA
HUNG.
ITALY

Motivation

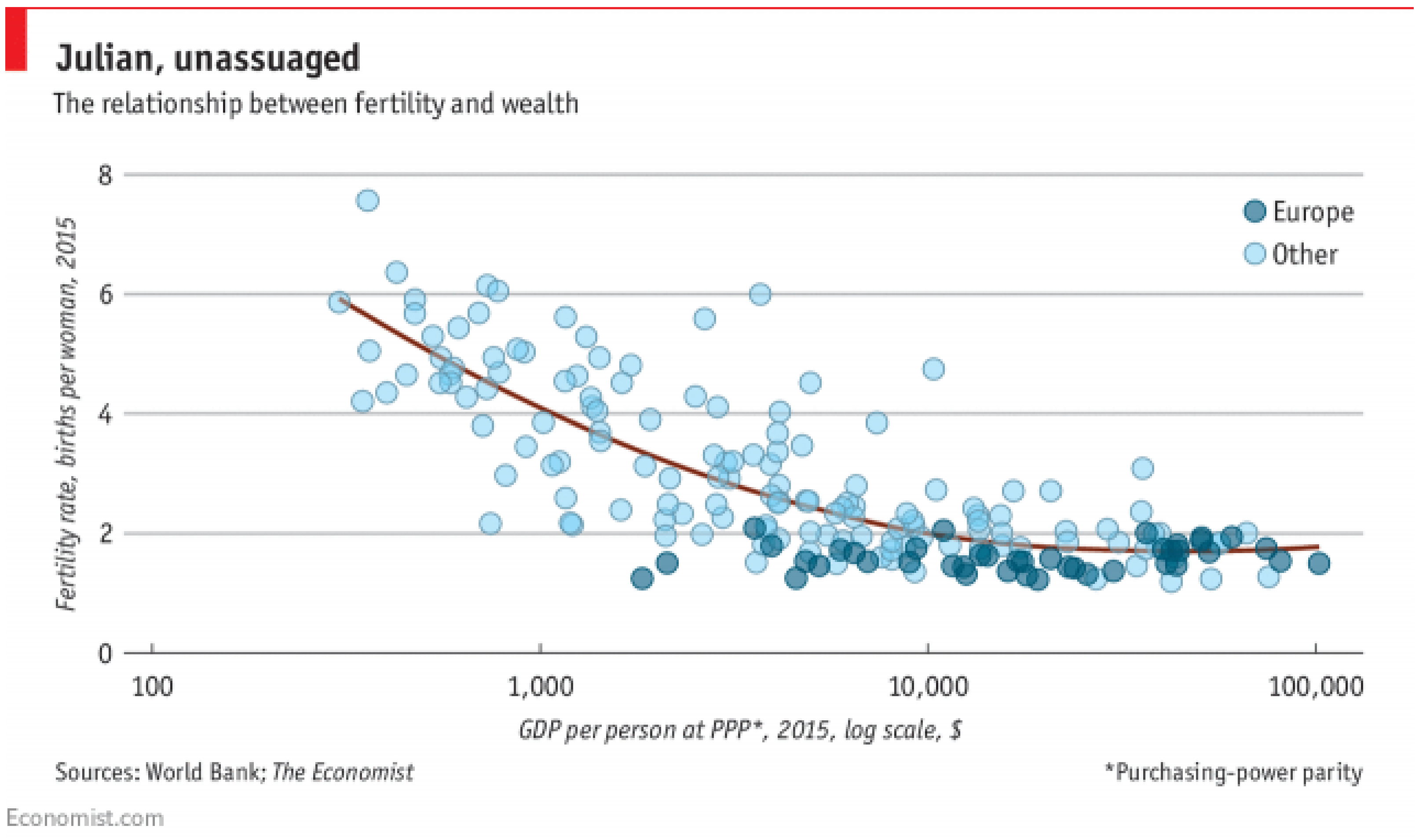
The truth is never linear!

Or almost never!





Wealth and Europe's low birth rates



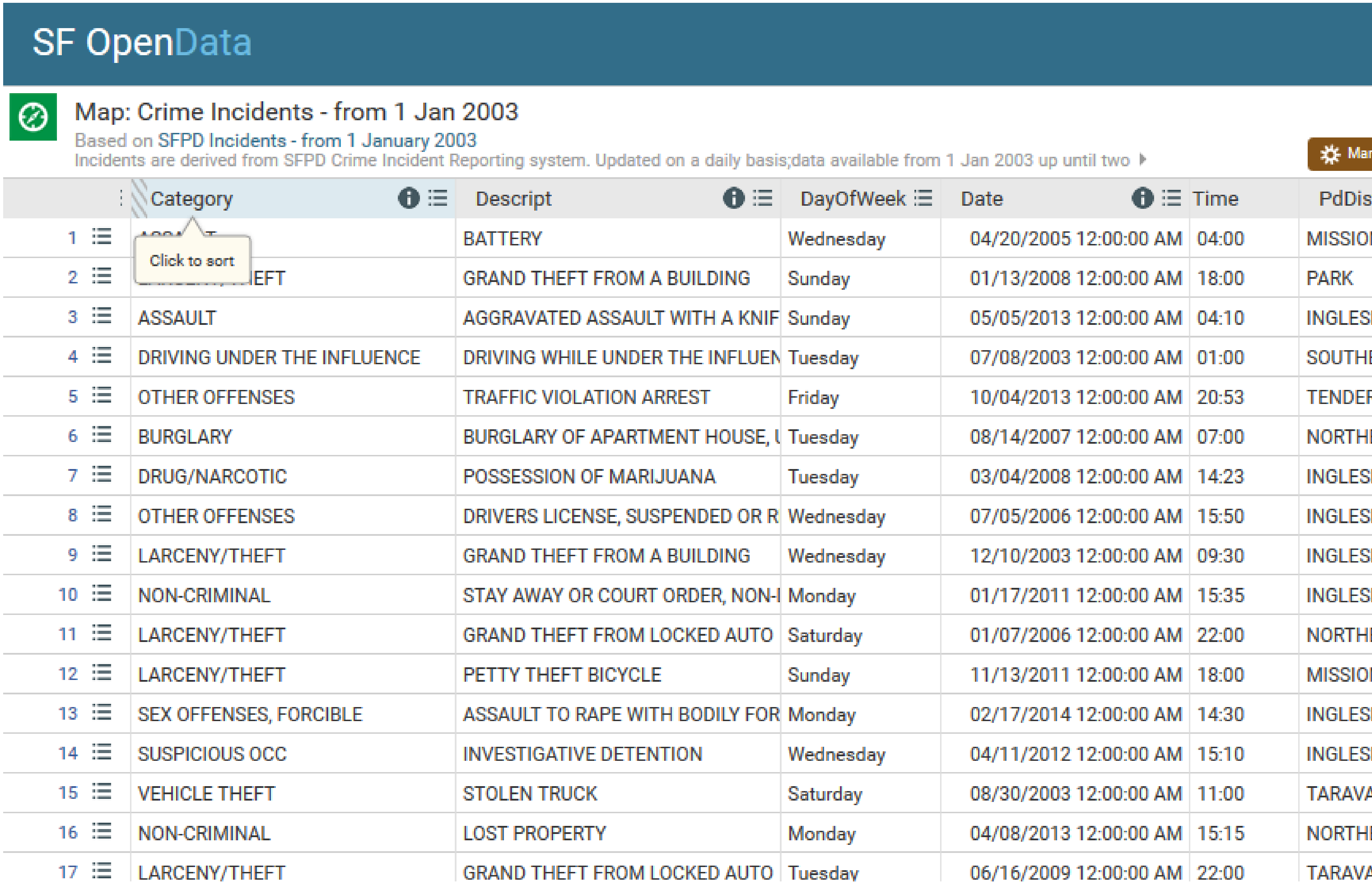
Birth rates are indeed highly correlated with national income. But the fertility rates of many European countries are lower than would be expected if GDP per person were the only factor that mattered. Romania, for instance, has 1.5 births per adult woman. Based purely on its level of economic development, that figure would be expected to be around 2.1.

Taxi Cab Animations

%

San Francisco Crime Data

https://data.sfgov.org/Public-Safety/Map-Crime-Incidents-from-1-Jan-2003/gxxq-x39z



SF OpenData

Map: Crime Incidents - from 1 Jan 2003

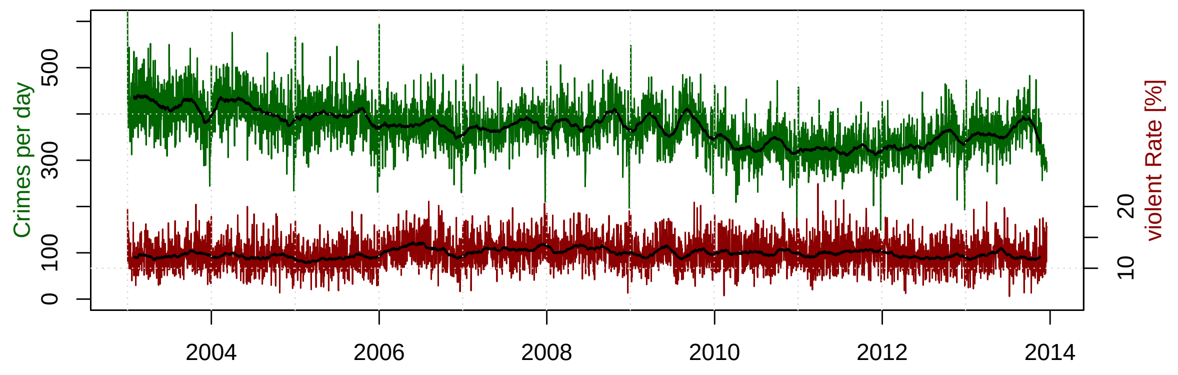
Based on SFPD Incidents - from 1 January 2003

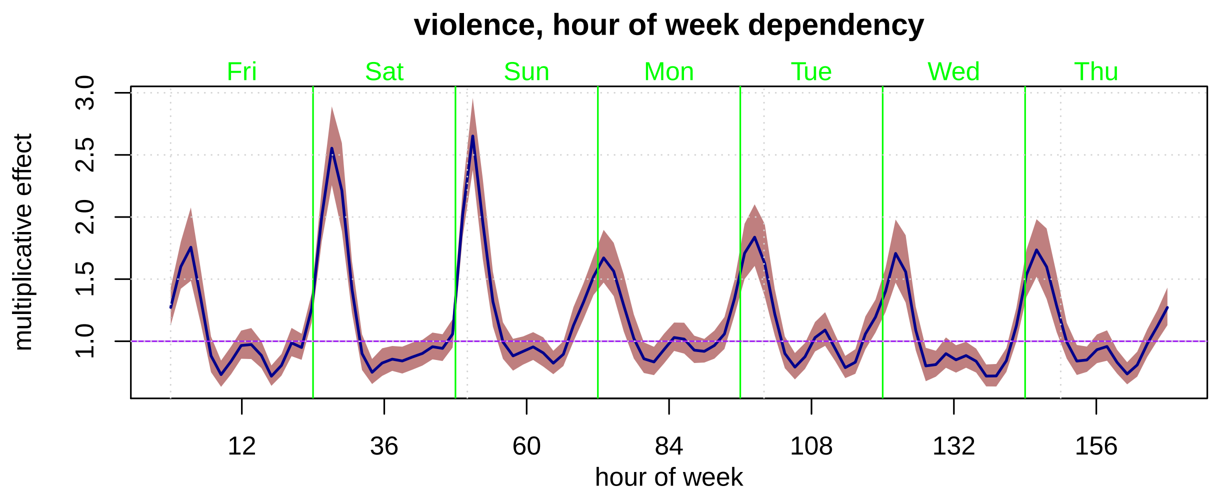
Incidents are derived from SFPD Crime Incident Reporting system. Updated on a daily basis;data available from 1 Jan 2003 up until two

Click to sort

	Category	Descript	DayOfWeek	Date	Time	PdDis
1	[illegible]	BATTERY	Wednesday	04/20/2005 12:00:00 AM	04:00	MISSIO
2	[illegible]EFT	GRAND THEFT FROM A BUILDING	Sunday	01/13/2008 12:00:00 AM	18:00	PARK
3	ASSAULT	AGGRAVATED ASSAULT WITH A KNIF	Sunday	05/05/2013 12:00:00 AM	04:10	INGLES
4	DRIVING UNDER THE INFLUENCE	DRIVING WHILE UNDER THE INFLUEN	Tuesday	07/08/2003 12:00:00 AM	01:00	SOUTHE
5	OTHER OFFENSES	TRAFFIC VIOLATION ARREST	Friday	10/04/2013 12:00:00 AM	20:53	TENDER
6	BURGLARY	BURGLARY OF APARTMENT HOUSE, U	Tuesday	08/14/2007 12:00:00 AM	07:00	NORTHE
7	DRUG/NARCOTIC	POSSESSION OF MARIJUANA	Tuesday	03/04/2008 12:00:00 AM	14:23	INGLES
8	OTHER OFFENSES	DRIVERS LICENSE, SUSPENDED OR R	Wednesday	07/05/2006 12:00:00 AM	15:50	INGLES
9	LARCENY/THEFT	GRAND THEFT FROM A BUILDING	Wednesday	12/10/2003 12:00:00 AM	09:30	INGLES
10	NON-CRIMINAL	STAY AWAY OR COURT ORDER, NON-I	Monday	01/17/2011 12:00:00 AM	15:35	INGLES
11	LARCENY/THEFT	GRAND THEFT FROM LOCKED AUTO	Saturday	01/07/2006 12:00:00 AM	22:00	NORTHE
12	LARCENY/THEFT	PETTY THEFT BICYCLE	Sunday	11/13/2011 12:00:00 AM	18:00	MISSIO
13	SEX OFFENSES, FORCIBLE	ASSAULT TO RAPE WITH BODILY FOR	Monday	02/17/2014 12:00:00 AM	14:30	INGLES
14	SUSPICIOUS OCC	INVESTIGATIVE DETENTION	Wednesday	04/11/2012 12:00:00 AM	15:10	INGLES
15	VEHICLE THEFT	STOLEN TRUCK	Saturday	08/30/2003 12:00:00 AM	11:00	TARAVA
16	NON-CRIMINAL	LOST PROPERTY	Monday	04/08/2013 12:00:00 AM	15:15	NORTHE
17	LARCENY/THEFT	GRAND THEFT FROM LOCKED AUTO	Tuesday	06/16/2009 12:00:00 AM	22:00	TARAVA

Temporal Analysis





Knowledge • 159 teams

San Francisco Crime Classification

Tue 2 Jun 2015

Nonlinear Modeling in R

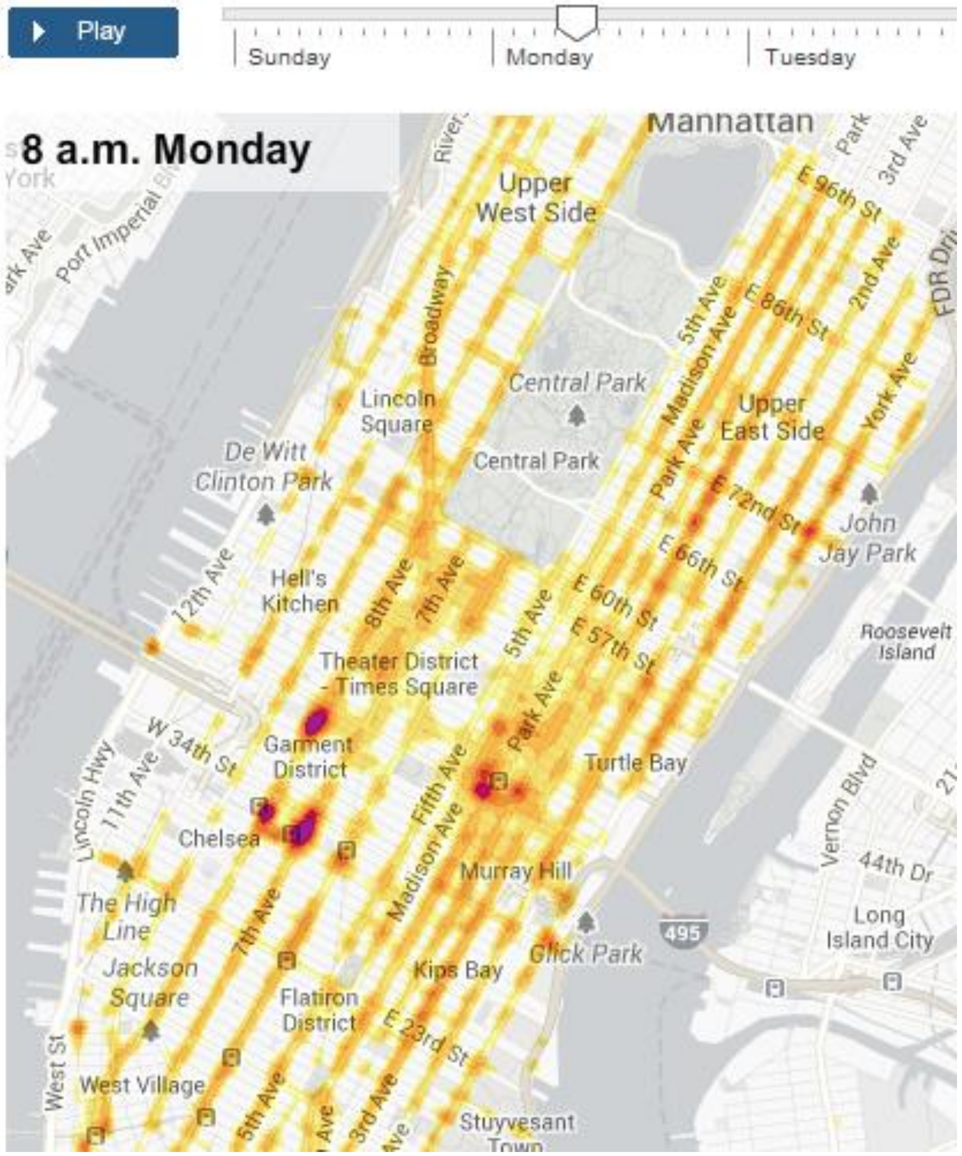
Expanding the R user‘s toolbox with splines, GAMs, trees and forests

Markus Löcher
Berlin School of Economics and Law

DENMARK
SWEDEN
North Sea
Baltic Sea
NETH.
U.K.
BERLIN
GERMANY
POLAND
BELG.
CZECH REP.
LUX.
SLVK.
FRANCE
SWITZ.
AUSTRIA
HUNG.
ITALY

NYC Cab Data

- 500k trips/day over 90 million trips in 6 months
- 30 GB raw data
- Hadoop for initial processing w/ MapReduce
- Map on key: gridcellid, weekhour



Moving Beyond Linearity

The truth is never linear!
Or almost never!

But often the linearity assumption is good enough.

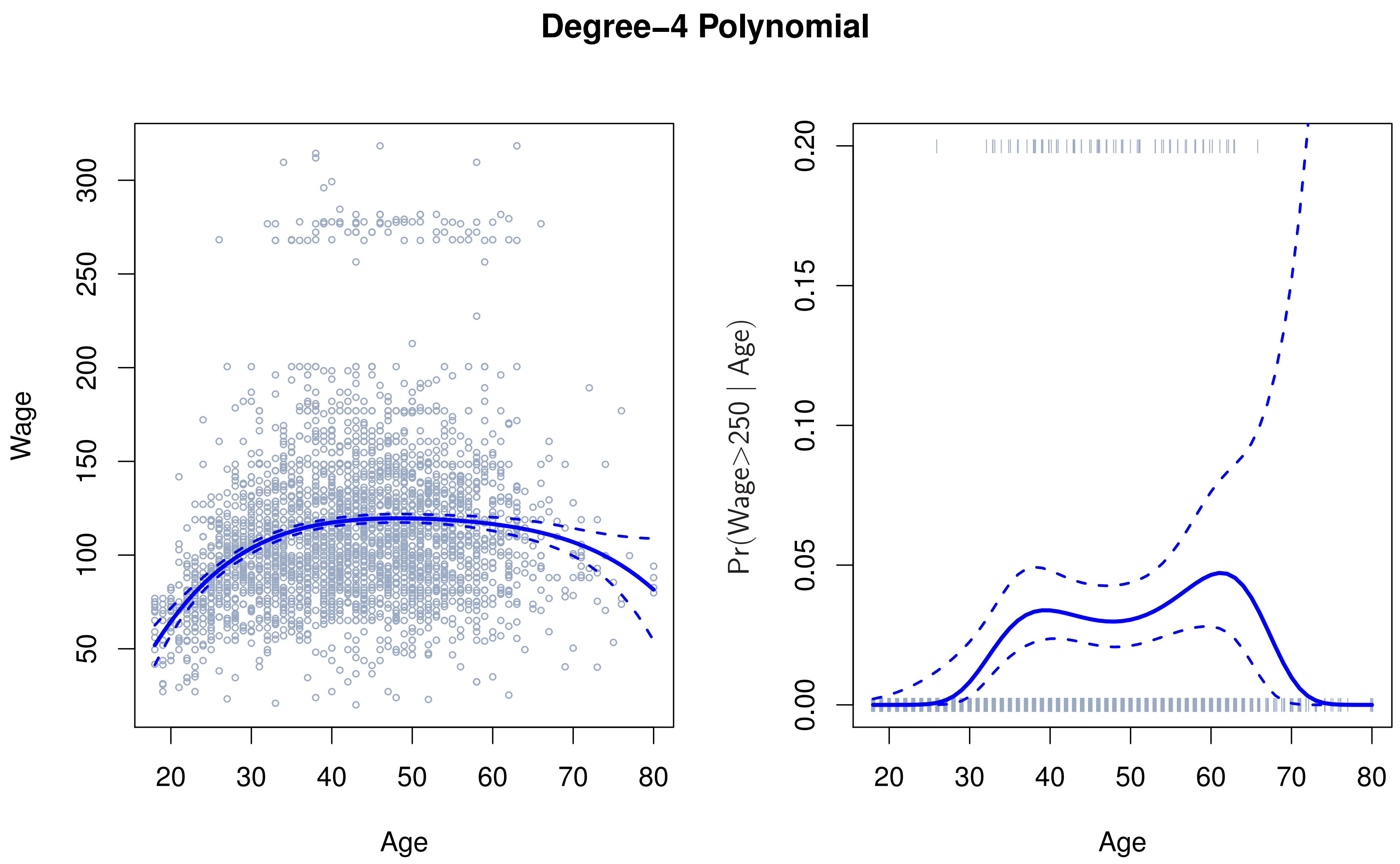
When its not . . .

- polynomials,
- step functions,
- splines,
- local regression, and
- generalized additive models

offer a lot of flexibility, without losing the ease and interpretability of linear models.

Polynomial Regression

$$y_i = \beta_0 + \beta_1 x_i + \beta_2 x_i^2 + \beta_3 x_i^3 + \ldots + \beta_d x_i^d + \epsilon_i$$



Details

- Create new variables $X_1 = X$, $X_2 = X^2$, etc and then treat as multiple linear regression.

Details

- Create new variables $X_1 = X$, $X_2 = X^2$, etc and then treat as multiple linear regression.
- Not really interested in the coefficients; more interested in the fitted function values at any value x_0:

$$\hat{f}(x_0) = \hat{\beta}_0 + \hat{\beta}_1 x_0 + \hat{\beta}_2 x_0^2 + \hat{\beta}_3 x_0^3 + \hat{\beta}_4 x_0^4.$$

Details

- Create new variables $X_1 = X$, $X_2 = X^2$, etc and then treat as multiple linear regression.
- Not really interested in the coefficients; more interested in the fitted function values at any value x_0:

$$\hat{f}(x_0) = \hat{\beta}_0 + \hat{\beta}_1 x_0 + \hat{\beta}_2 x_0^2 + \hat{\beta}_3 x_0^3 + \hat{\beta}_4 x_0^4.$$

- Since $\hat{f}(x_0)$ is a linear function of the $\hat{\beta}_\ell$, can get a simple expression for *pointwise-variances* $\mathrm{Var}[\hat{f}(x_0)]$ at any value x_0. In the figure we have computed the fit and pointwise standard errors on a grid of values for x_0. We show $\hat{f}(x_0) \pm 2 \cdot \mathrm{se}[\hat{f}(x_0)]$.

Details

- Create new variables $X_1 = X$, $X_2 = X^2$, etc and then treat as multiple linear regression.
- Not really interested in the coefficients; more interested in the fitted function values at any value x_0:

$$\hat{f}(x_0) = \hat{\beta}_0 + \hat{\beta}_1 x_0 + \hat{\beta}_2 x_0^2 + \hat{\beta}_3 x_0^3 + \hat{\beta}_4 x_0^4.$$

- Since $\hat{f}(x_0)$ is a linear function of the $\hat{\beta}_\ell$, can get a simple expression for *pointwise-variances* $\text{Var}[\hat{f}(x_0)]$ at any value x_0. In the figure we have computed the fit and pointwise standard errors on a grid of values for x_0. We show $\hat{f}(x_0) \pm 2 \cdot \text{se}[\hat{f}(x_0)]$.
- We either fix the degree d at some reasonably low value, else use cross-validation to choose d.

Details continued

- Logistic regression follows naturally. For example, in figure we model

$$\Pr(y_i > 250 | x_i) = \frac{\exp(\beta_0 + \beta_1 x_i + \beta_2 x_i^2 + \ldots + \beta_d x_i^d)}{1 + \exp(\beta_0 + \beta_1 x_i + \beta_2 x_i^2 + \ldots + \beta_d x_i^d)}.$$

- To get confidence intervals, compute upper and lower bounds on *on the logit scale*, and then invert to get on probability scale.

Details continued

- Logistic regression follows naturally. For example, in figure we model

$$\Pr(y_i > 250|x_i) = \frac{\exp(\beta_0 + \beta_1 x_i + \beta_2 x_i^2 + \ldots + \beta_d x_i^d)}{1 + \exp(\beta_0 + \beta_1 x_i + \beta_2 x_i^2 + \ldots + \beta_d x_i^d)}.$$

- To get confidence intervals, compute upper and lower bounds on *on the logit scale*, and then invert to get on probability scale.
- Can do separately on several variables—just stack the variables into one matrix, and separate out the pieces afterwards (see GAMs later).

Details continued

- Logistic regression follows naturally. For example, in figure we model

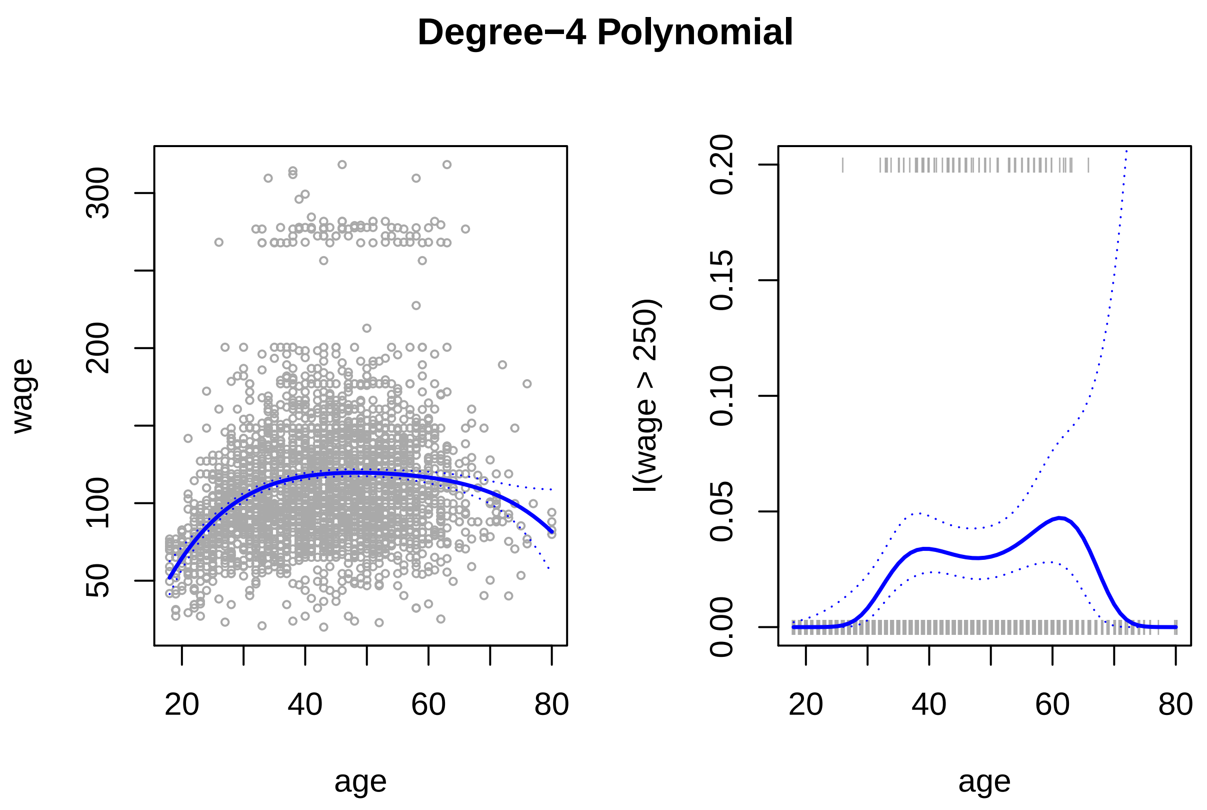
$$\Pr(y_i > 250|x_i) = \frac{\exp(\beta_0 + \beta_1 x_i + \beta_2 x_i^2 + \ldots + \beta_d x_i^d)}{1 + \exp(\beta_0 + \beta_1 x_i + \beta_2 x_i^2 + \ldots + \beta_d x_i^d)}.$$

- To get confidence intervals, compute upper and lower bounds on *on the logit scale*, and then invert to get on probability scale.
- Can do separately on several variables—just stack the variables into one matrix, and separate out the pieces afterwards (see GAMs later).
- Caveat: polynomials have notorious tail behavior — very bad for extrapolation.
- Can fit using $\texttt{y} \sim \texttt{poly}(\texttt{x}, \texttt{degree} = \texttt{3})$ in formula.

Polynomial Regression

Code to produce Figure 7.1:

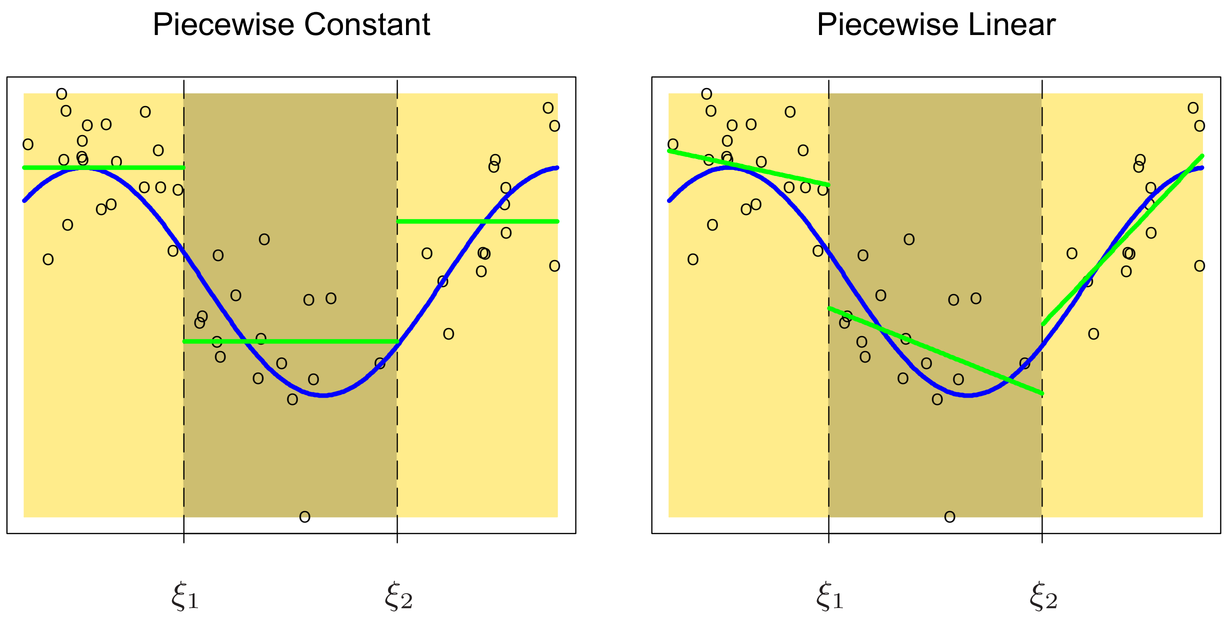
```r
library(ISLR)
attach(Wage)

fit <- lm(wage ~ poly(age, 4), data = Wage)
#coef(summary(fit))

fit2 <- lm(wage ~ poly(age, 4, raw =TRUE), data = Wage)
#coef(summary(fit2))

agelims <- range(age)
age.grid <- seq(from = agelims[1], to = agelims[2])
preds <- predict(fit, newdata = list(age = age.grid), se = TRUE)
se.bands <- cbind(preds$fit + 2*preds$se.fit, preds$fit - 2*preds$se.fit)

par(mfrow = c(1, 2), mar = c(4.5, 4.5, 1, 1), oma = c(0, 0, 4, 0))
plot(age, wage, xlim = agelims, cex = 0.5, col = "darkgrey")
title("Degree-4 Polynomial", outer = TRUE)
lines(age.grid, preds$fit, lwd = 2, col = "blue")
matlines(age.grid, se.bands, lwd = 1, col = "blue", lty = 3)

fitLR <- glm(I(wage > 250) ~ poly(age, 4), data = Wage, family = binomial)

preds <- predict(fitLR, newdata = list(age = age.grid), se = TRUE)

pfit <- exp(preds$fit)/(1 + exp(preds$fit))
se.bands.logit <- cbind(preds$fit + 2*preds$se.fit, preds$fit - 2*preds$se.fit)
se.bands <- exp(se.bands.logit)/(1 + exp(se.bands.logit))

preds <- predict(fitLR, newdata = list(age = age.grid), type = "response", se = TRUE)

plot(age, I(wage > 250), xlim = agelims, type = "n", ylim = c(0, 0.2))
points(jitter(age), I((wage > 250)/5), cex = 0.5, pch = "|", col = "darkgrey")
lines(age.grid, pfit, lwd = 2, col = "blue")
matlines(age.grid, se.bands, lwd = 1, col = "blue", lty = 3)
```



1. **Exercises**
   (a) Compare fits using `poly` with "manual" polynomials
   (b) Use `anova` to find the best degree
   (c) Produce an analogous plot for the *Auto* data.

Domain Partitions

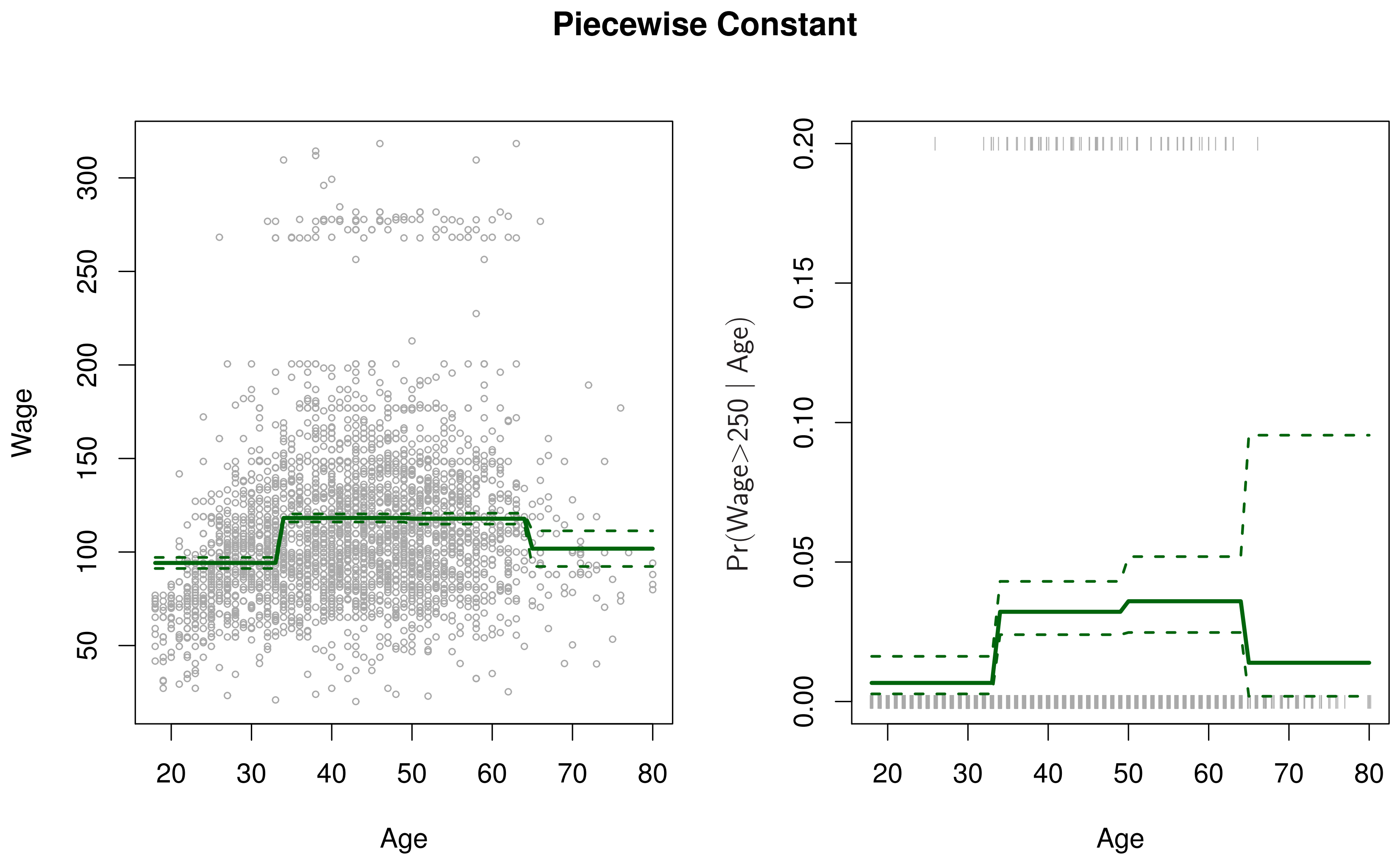
Using polynomial functions of the features as predictors in a linear model imposes a **global** structure on the non-linear function of X. We can instead adapt a more **local** approach and use step functions by breaking the range of X into bins, and fit a different constant in each bin. This amounts to converting a continuous variable into an *ordered categorical variable*. (We will see similar ideas with trees later on)



Step Functions

Another way of creating transformations of a variable — cut the variable into distinct regions.

$$C_1(X) = I(X < 35), \quad C_2(X) = I(35 \le X < 50), \ldots, C_3(X) = I(X \ge 65)$$



Step functions continued

- Easy to work with. Creates a series of dummy variables representing each group.

Step functions continued

- Easy to work with. Creates a series of dummy variables representing each group.
- Useful way of creating interactions that are easy to interpret. For example, interaction effect of `Year` and `Age`:

$$I(\texttt{Year} < 2005) \cdot \texttt{Age}, \quad I(\texttt{Year} \geq 2005) \cdot \texttt{Age}$$

would allow for different linear functions in each age category.

Step functions continued

- Easy to work with. Creates a series of dummy variables representing each group.
- Useful way of creating interactions that are easy to interpret. For example, interaction effect of `Year` and `Age`:

$$I(\texttt{Year} < 2005) \cdot \texttt{Age}, \quad I(\texttt{Year} \geq 2005) \cdot \texttt{Age}$$

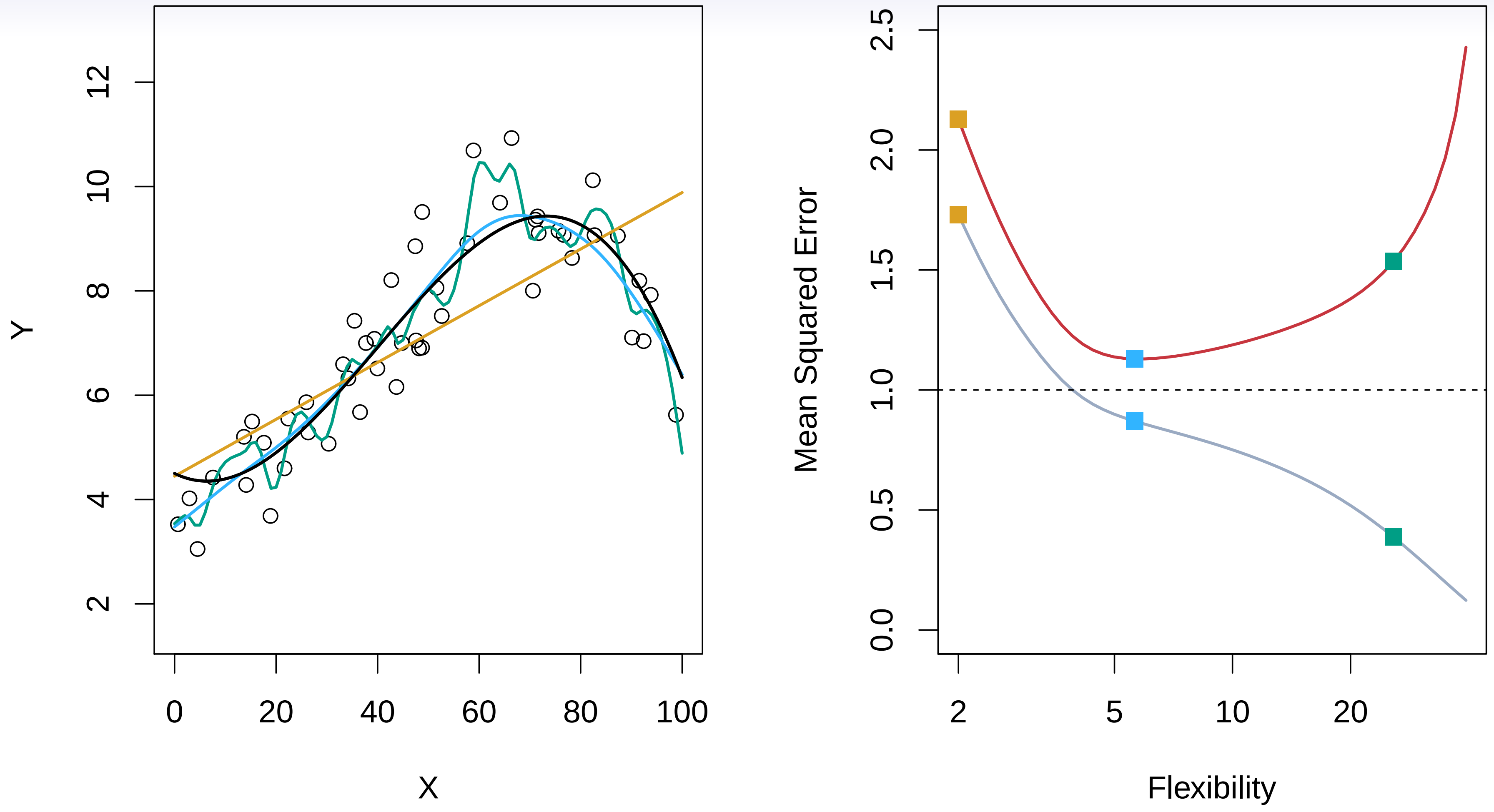
  would allow for different linear functions in each age category.
- In R: `I(year < 2005)` or `cut(age, c(18, 25, 40, 65, 90))`.

Step functions continued

- Easy to work with. Creates a series of dummy variables representing each group.
- Useful way of creating interactions that are easy to interpret. For example, interaction effect of `Year` and `Age`:

$$I(\texttt{Year} < 2005) \cdot \texttt{Age}, \quad I(\texttt{Year} \geq 2005) \cdot \texttt{Age}$$

  would allow for different linear functions in each age category.
- In R: `I(year < 2005)` or `cut(age, c(18, 25, 40, 65, 90))`.
- Choice of cutpoints or *knots* can be problematic. For creating nonlinearities, smoother alternatives such as *splines* are available.



Black curve is truth. Red curve on right is $\text{MSE}_{\textsf{Te}}$, grey curve is $\text{MSE}_{\textsf{Tr}}$. Orange, blue and green curves/squares correspond to fits of different flexibility.

Bias-Variance Trade-off

Suppose we have fit a model $\hat{f}(x)$ to some training data Tr, and let (x_0, y_0) be a test observation drawn from the population. If the true model is $Y = f(X) + \epsilon$ (with $f(x) = E(Y|X = x)$), then

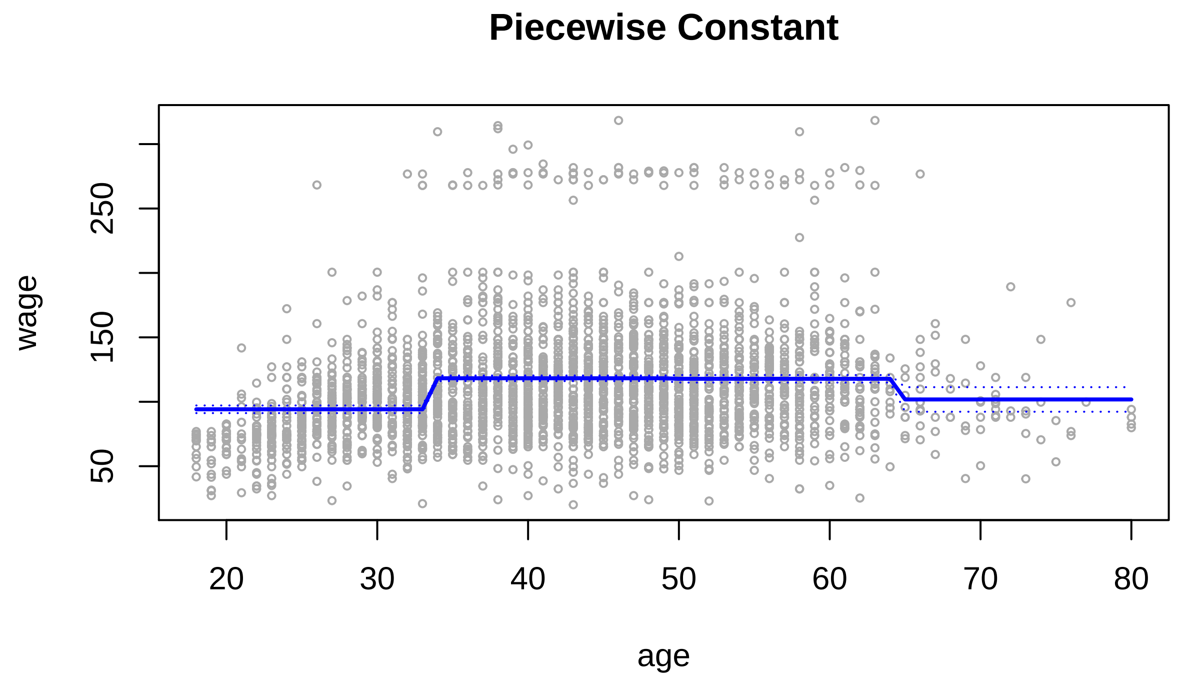
$$E\left(y_0 - \hat{f}(x_0)\right)^2 = \text{Var}(\hat{f}(x_0)) + [\text{Bias}(\hat{f}(x_0))]^2 + \text{Var}(\epsilon).$$

The expectation averages over the variability of y_0 as well as the variability in Tr. Note that $\text{Bias}(\hat{f}(x_0))] = E[\hat{f}(x_0)] - f(x_0)$.

Typically as the *flexibility* of $\hat{f}$ increases, its variance increases, and its bias decreases. So choosing the flexibility based on average test error amounts to a *bias-variance trade-off.*

Code to produce Figure 7.2

```
fit <- lm(wage ~ cut(age, 4), data = Wage)
preds <- predict(fit, newdata = list(age = age.grid), se = TRUE)
se.bands <- cbind(preds$fit + 2*preds$se.fit, preds$fit - 2*preds$se.fit)

plot(age, wage, xlim = agelims, cex = 0.5, col = "darkgrey", main = "Piecewise Constant")
lines(age.grid, preds$fit, lwd = 2, col = "blue")
matlines(age.grid, se.bands, lwd = 1, col = "blue", lty = 3)
```



2. **Exercises**

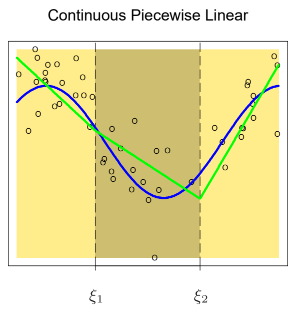
   Complete the 2nd plot

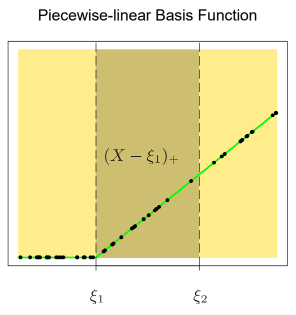
Basis Functions

Polynomial and piecewise-constant regression models are in fact special cases of a **basis function** approach. The idea is to have at hand a family of functions or transformations that can be applied to a variable X. Instead of fitting a linear model in X, we fit the model

$$y_i = \beta_0 + \beta_1 b_1(x_i) + \beta_2 b_2(x_i) \ldots + \beta_K b_K(x_i) + \epsilon_i$$

We can think of (2) as a standard linear model with predictors $b_1(x_i), \ldots, b_K(x_i)$. Hence, we can use least squares to estimate the unknown regression coefficients. Importantly, this means that all of the inference tools for linear models, such as standard errors for the coefficient estimates and F-statistics for the model's overall significance, are available in this setting.





Regression Splines

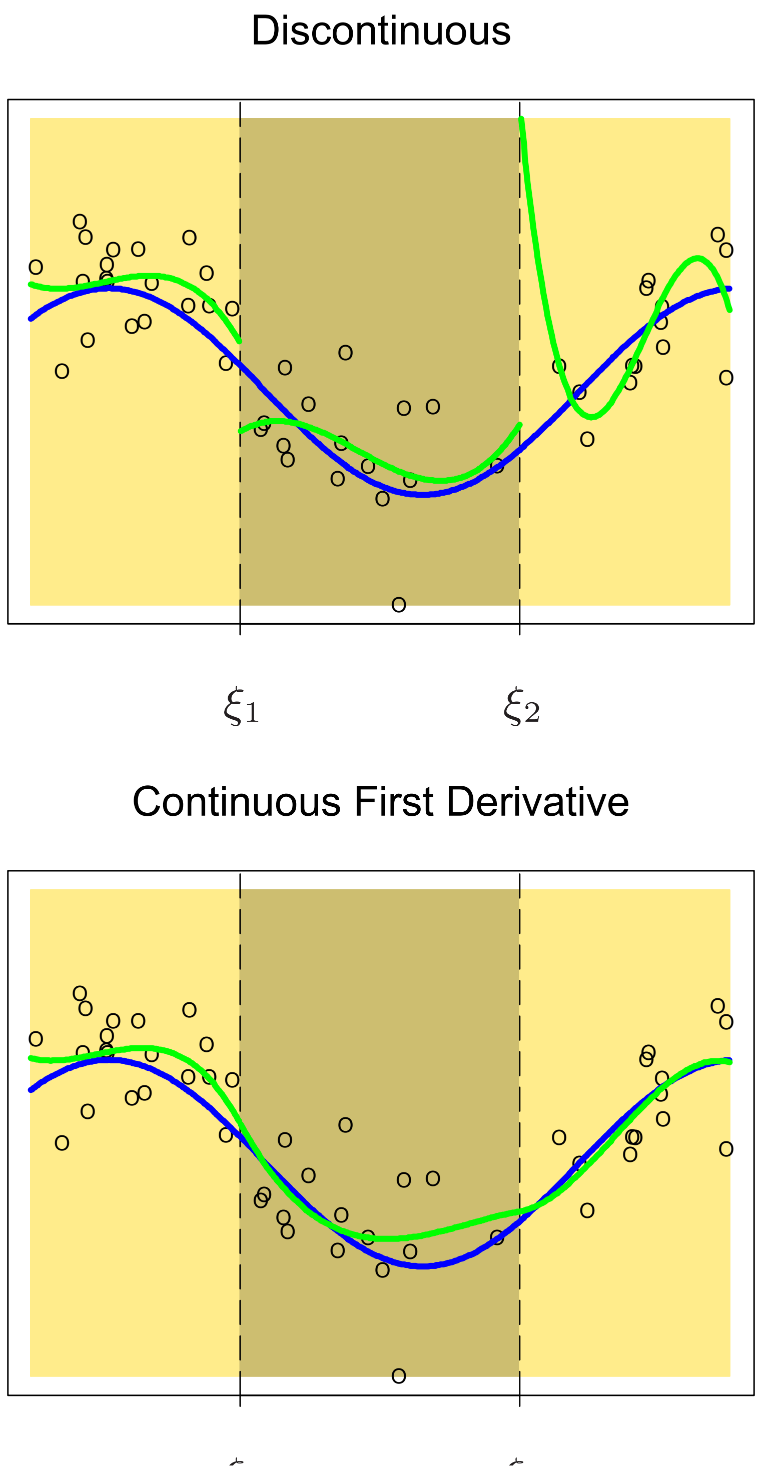
Piecewise Polynomials

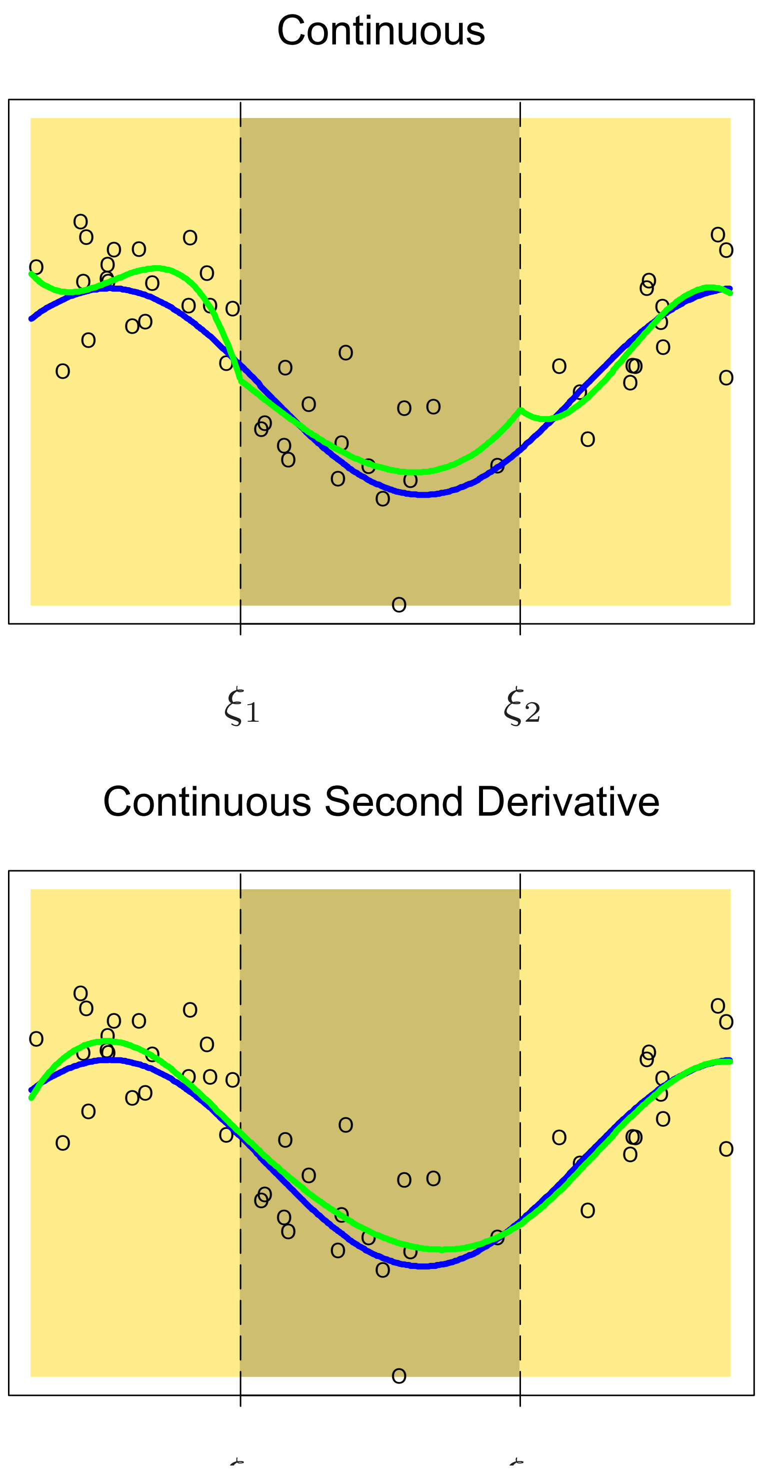
Instead of fitting a high-degree polynomial over the entire range of X, piecewise polynomial regression involves fitting separate low-degree polynomials over different regions of X. The points where the coefficients change are called **knots**.

A piecewise cubic polynomial with a single knot at a point c takes the form

$$y_i = \begin{cases} \beta_{01} + \beta_{11}x_i + \beta_{21}x_i^2 + \beta_{31}x_i^3 + \epsilon_i, & \text{if } x \leq c \\ \beta_{02} + \beta_{12}x_i + \beta_{22}x_i^2 + \beta_{32}x_i^3 + \epsilon_i, & \text{if } x \geq c \end{cases}$$

Using more knots leads to a more flexible piecewise polynomial. In general, if we place K different knots throughout the range of X, then we will end up fitting K + 1 different cubic polynomials. Note that we do not need to use a cubic polynomial. For example, we can instead fit piecewise linear functions. In fact, our piecewise constant functions from above are piecewise polynomials of degree 0!


Discontinuous
ξ_1
ξ_2
Continuous First Derivative

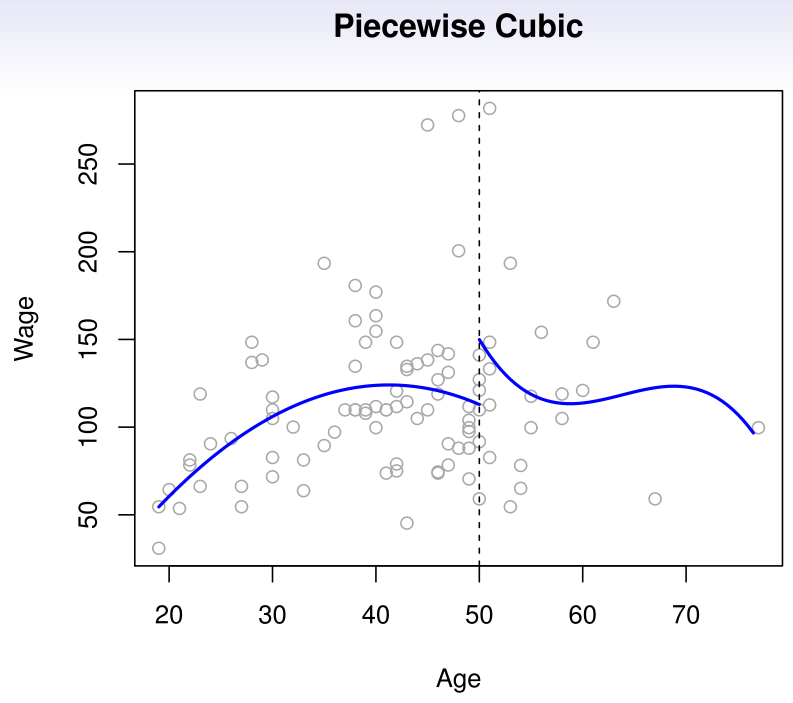

Continuous
ξ_1
ξ_2
Continuous Second Derivative

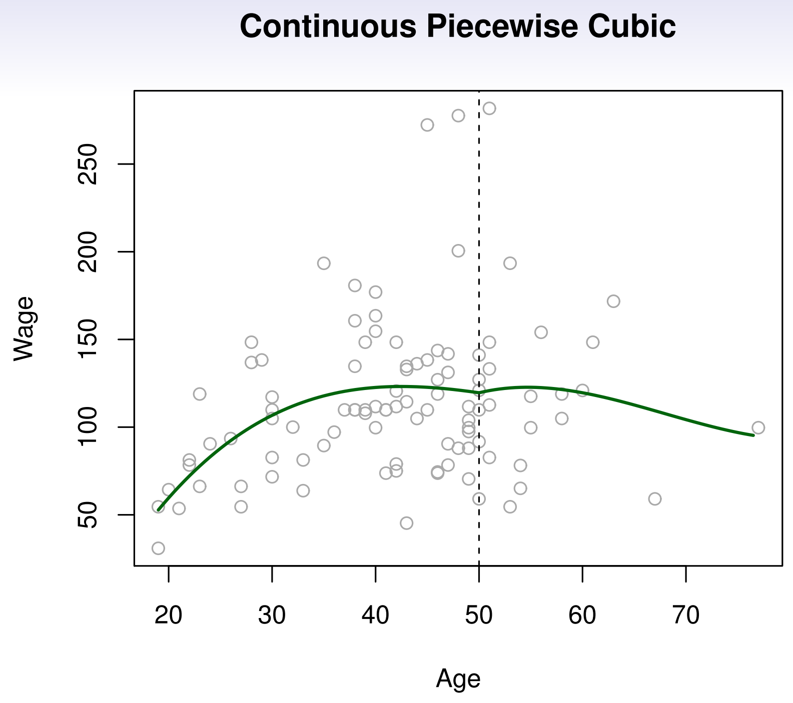
Piecewise Polynomials

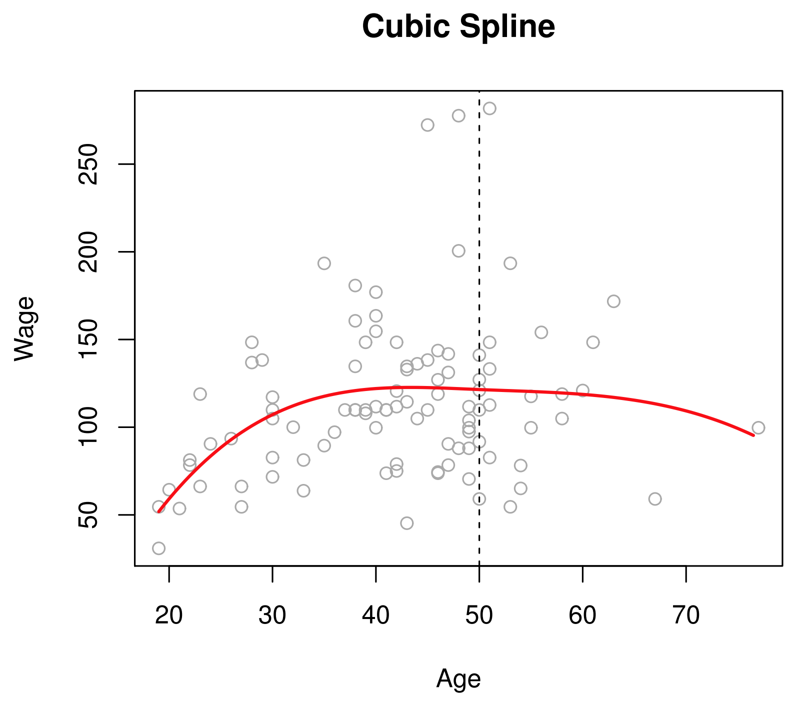
- Instead of a single polynomial in X over its whole domain, we can rather use different polynomials in regions defined by knots. E.g. (see figure)

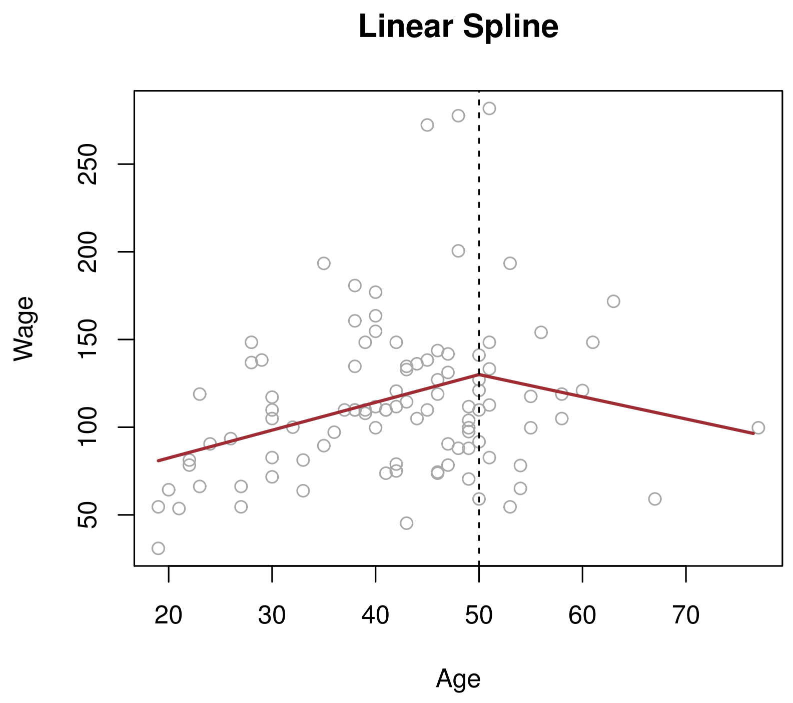
$$y_i = \begin{cases} \beta_{01} + \beta_{11}x_i + \beta_{21}x_i^2 + \beta_{31}x_i^3 + \epsilon_i & \text{if } x_i < c; \\ \beta_{02} + \beta_{12}x_i + \beta_{22}x_i^2 + \beta_{32}x_i^3 + \epsilon_i & \text{if } x_i \geq c. \end{cases}$$

- Better to add constraints to the polynomials, e.g. continuity.
- *Splines* have the "maximum" amount of continuity.


Piecewise Cubic
Wage
Age
50
100
150
200
250
20
30
40
50
60
70


Continuous Piecewise Cubic
Wage
Age
50
100
150
200
250
20
30
40
50
60
70


Cubic Spline
Wage
Age
50
100
150
200
250
20
30
40
50
60
70


Linear Spline
Wage
Age
50
100
150
200
250
20
30
40
50
60
70

Linear Splines

A linear spline with knots at ξ_k, $k = 1, \ldots, K$ is a piecewise linear polynomial continuous at each knot.

We can represent this model as

$$y_i = \beta_0 + \beta_1 b_1(x_i) + \beta_2 b_2(x_i) + \cdots + \beta_{K+3} b_{K+3}(x_i) + \epsilon_i,$$

where the b_k are *basis functions*.

Linear Splines

A linear spline with knots at ξ_k, $k = 1, \ldots, K$ is a piecewise linear polynomial continuous at each knot.

We can represent this model as

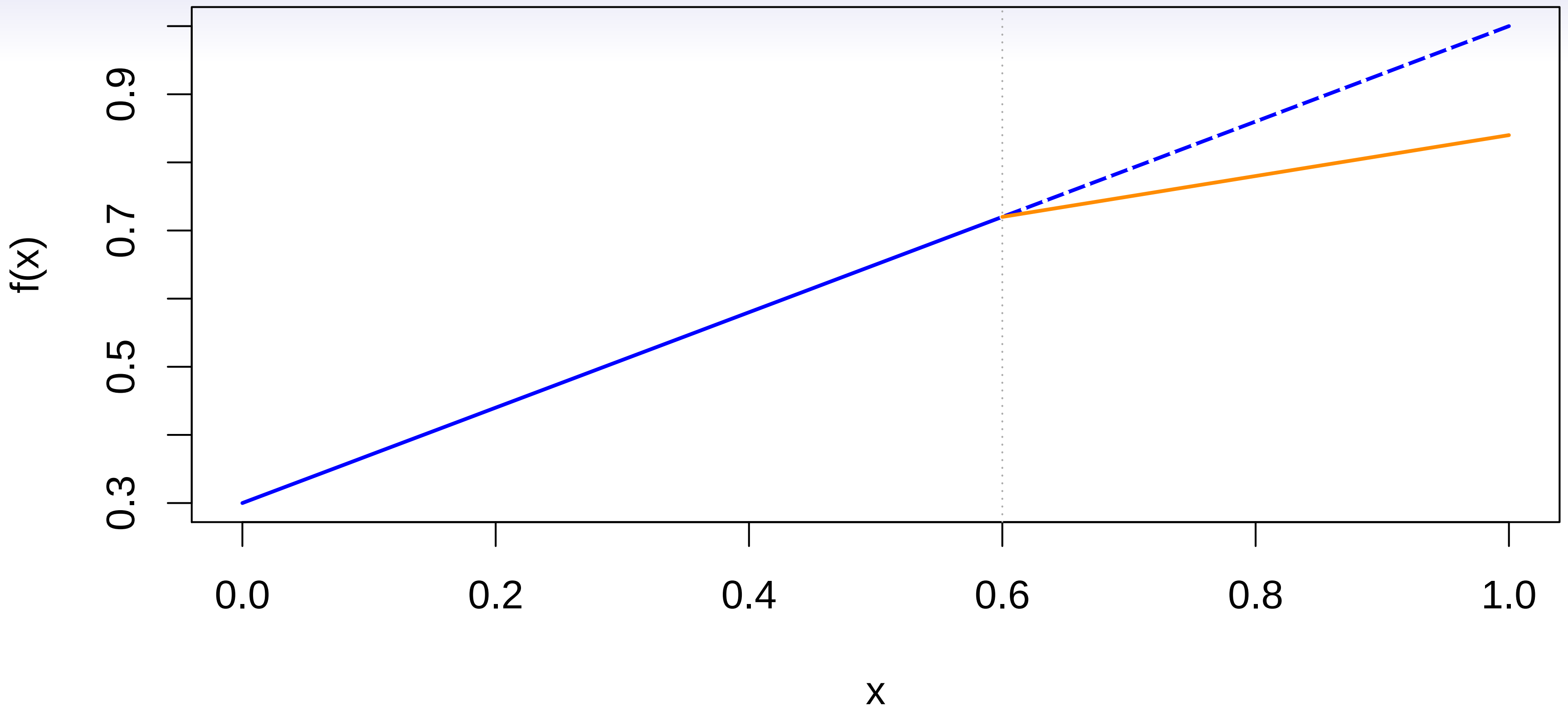
$$y_i = \beta_0 + \beta_1 b_1(x_i) + \beta_2 b_2(x_i) + \cdots + \beta_{K+3} b_{K+3}(x_i) + \epsilon_i,$$

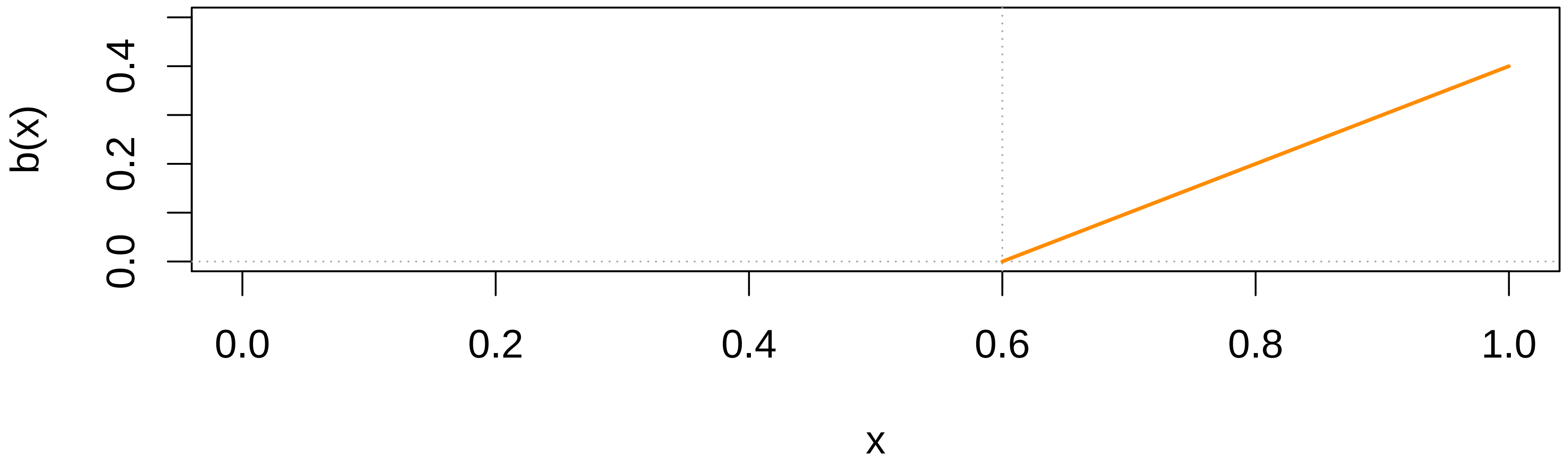
where the b_k are *basis functions*.

$$\begin{aligned} b_1(x_i) &= x_i \\ b_{k+1}(x_i) &= (x_i - \xi_k)_+, \quad k = 1, \ldots, K \end{aligned}$$

Here the $()_+$ means *positive part*; i.e.

$$(x_i - \xi_k)_+ = \begin{cases} x_i - \xi_k & \text{if } x_i > \xi_k \\ 0 & \text{otherwise} \end{cases}$$


f(x)
0.3
0.5
0.7
0.9
0.0
0.2
0.4
0.6
0.8
1.0
x


b(x)
0.0
0.2
0.4
0.0
0.2
0.4
0.6
0.8
1.0
x

Cubic Splines

A cubic spline with knots at ξ_k, $k = 1, \dots, K$ is a piecewise cubic polynomial with continuous derivatives up to order 2 at each knot.

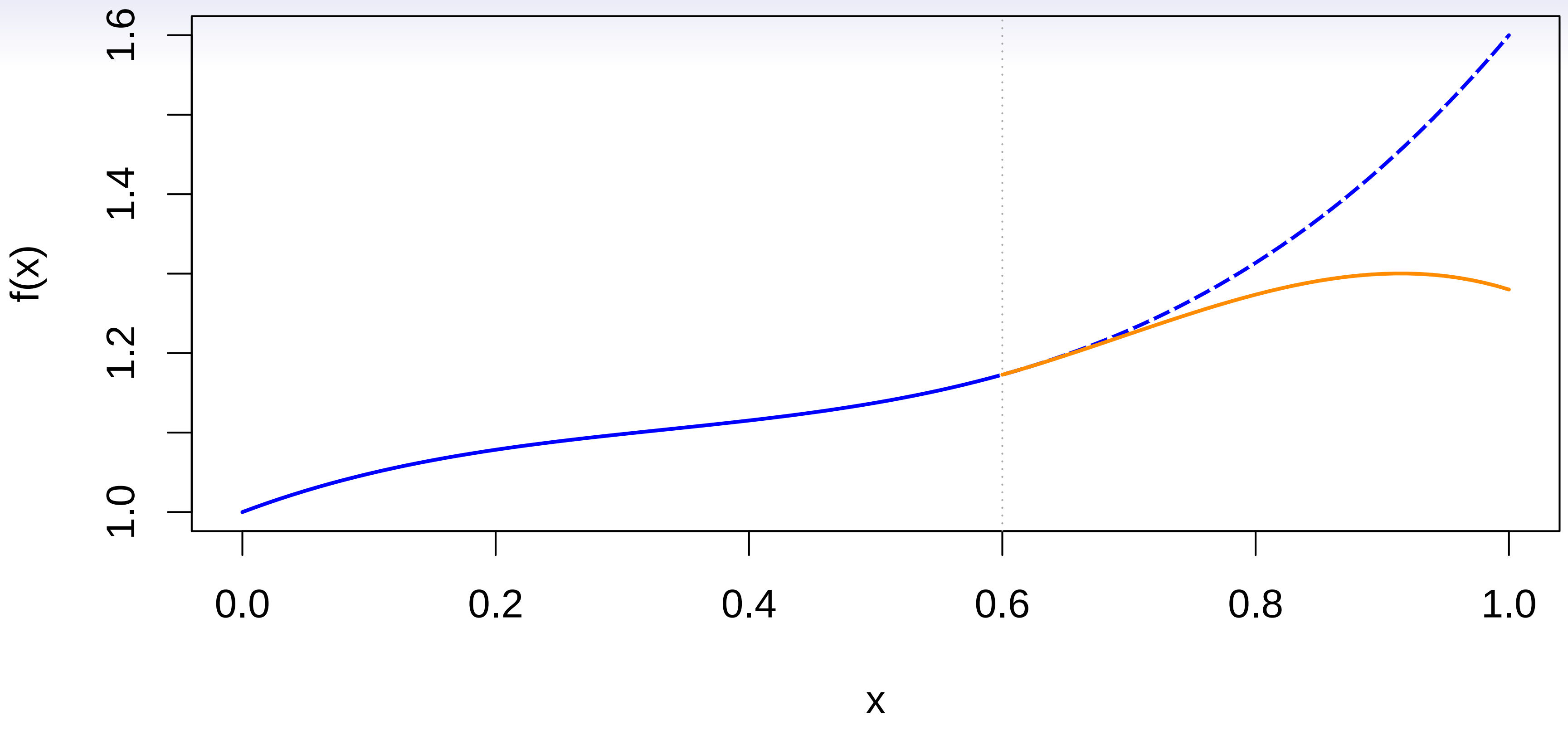
Again we can represent this model with truncated power basis functions

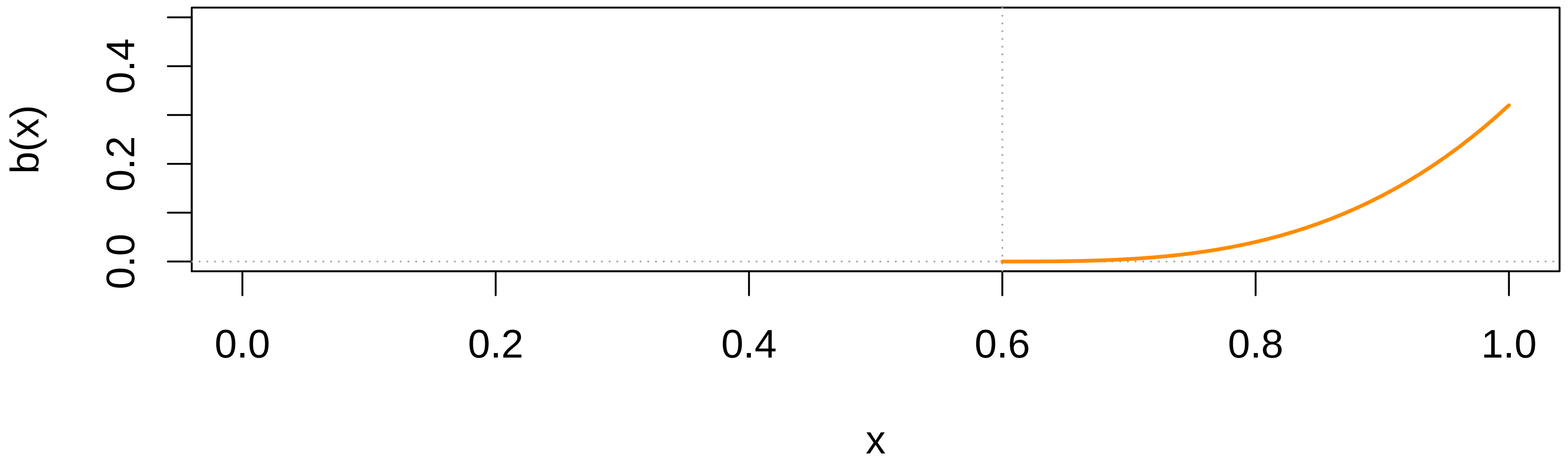
$$y_i = \beta_0 + \beta_1 b_1(x_i) + \beta_2 b_2(x_i) + \cdots + \beta_{K+3} b_{K+3}(x_i) + \epsilon_i,$$

$$\begin{aligned} b_1(x_i) &= x_i \\ b_2(x_i) &= x_i^2 \\ b_3(x_i) &= x_i^3 \\ b_{k+3}(x_i) &= (x_i - \xi_k)_+^3, \quad k = 1, \dots, K \end{aligned}$$

where

$$(x_i - \xi_k)_+^3 = \begin{cases} (x_i - \xi_k)^3 & \text{if } x_i > \xi_k \\ 0 & \text{otherwise} \end{cases}$$


f(x)
1.0
1.2
1.4
1.6
0.0
0.2
0.4
0.6
0.8
1.0
x

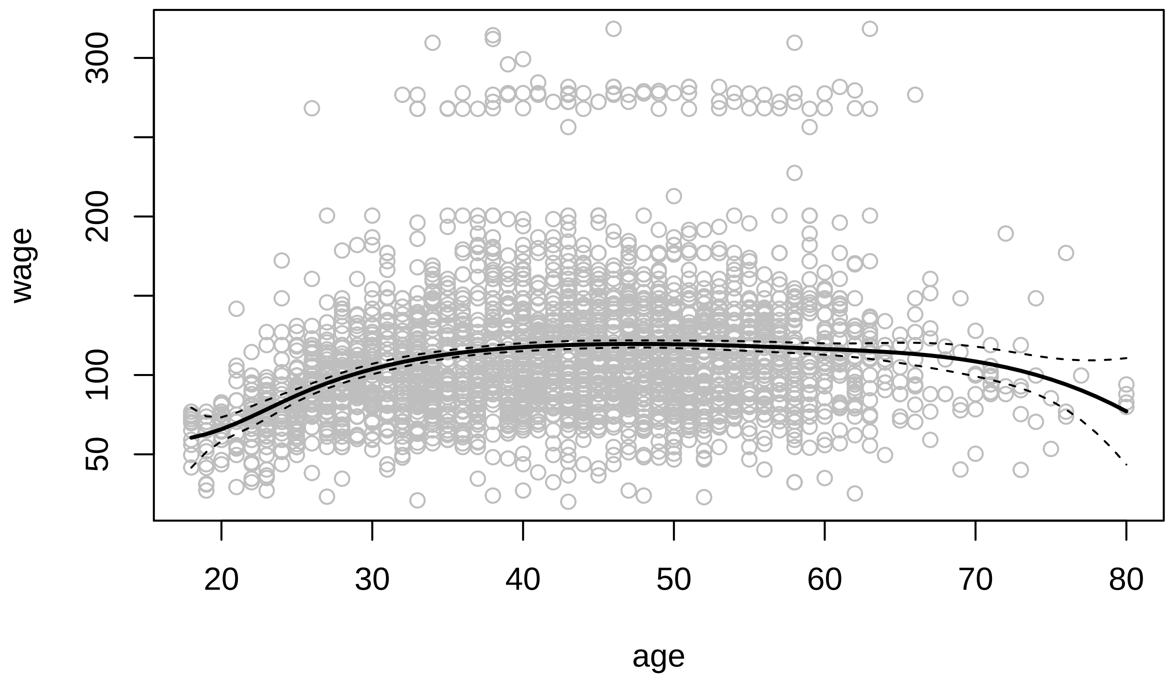

b(x)
0.0
0.2
0.4
0.0
0.2
0.4
0.6
0.8
1.0
x

Regression Splines in R

Fitting splines in R is easy: `bs(x, ...)` for any degree splines, and `ns(x, ...)` for natural cubic splines, in package splines.

In order to fit regression splines in R, we use the splines library. We saw that regression splines can be fit by constructing an appropriate matrix of basis functions. The bs() function generates the entire matrix of basis functions for splines with the specified set of knots. By default, cubic splines are produced. Fitting wage to age using a regression spline is simple

```
agelims=range(age)
age.grid=seq(from=agelims[1],to=agelims[2])



fit=lm(wage~bs(age,knots=c(25,40,60)),data=Wage)
pred=predict(fit,newdata=list(age=age.grid),se=T)
plot(age,wage,col="gray")
lines(age.grid,pred$fit,lwd=2)
lines(age.grid,pred$fit+2*pred$se,lty="dashed")
lines(age.grid,pred$fit-2*pred$se,lty="dashed")
```



Here we have prespecified knots at ages 25, 40, and 60. This produces a spline with six basis functions. (Recall that a cubic spline with three knots has seven degrees of freedom; these degrees of freedom are used up by an intercept, plus six basis functions.) We could also use the df option to produce a spline with knots at uniform quantiles of the data.

```
dim(bs(age,knots=c(25,40,60)))
```

```
## [1] 3000    6
```

```
dim(bs(age,df=6))
```

```
## [1] 3000    6
```

```
attr(bs(age,df=6),"knots")
```

```
##   25%   50%   75% 
## 33.75 42.00 51.00
```

In this case R chooses knots at ages 33.8, 42.0, and 51.0, which correspond to the 25th, 50th, and 75th percentiles of age. The function bs() also has a degree argument, so we can fit splines of any degree, rather than the default degree of 3 (which yields a cubic spline). In order to instead fit a natural spline, we use the ns() function. Here we fit a natural spline with four degrees of freedom.

```
fit2=lm(wage~ns(age,df=4),data=Wage)
pred2=predict(fit2,newdata=list(age=age.grid),se=T)
#plot(age,wage,xlim=agelims,cex=.5,col="darkgrey")
#lines(age.grid, pred2$fit,col="red",lwd=2)
```

A cubic spline with K knots uses a total of ?? degrees of freedom.

3. **Exercises, Coding**

   Fit cubic and natural cubic splines to the motorcycle data

   (a) with prespecified knots

   (b) with knots at uniform quantiles of the data

   (c) How would you decide on the optimal knots ?

4. **Exercises, Conceptual**

   It was mentioned above that a cubic regression spline with one knot at ξ can be obtained using a basis of the form x; x^2, x^3, $(x-\xi)^3_+$, where $(x-\xi)^3_+ = (x-\xi)^3$ if $x > \xi$ and equals 0 otherwise. We will now show that a function of the form

   $$f(x) = \beta_0 + \beta_1 x + \beta_2 x^2 + \beta_3 x^3 + \beta_4 (x-\xi)^3_+$$

   is indeed a cubic regression spline, regardless of the values of $\beta_0, \beta_1, \beta_2, \beta_3, \beta_4$.

   (a) Find a cubic polynomial

   $$f_1(x) = a_1 + b_1 x + c_1 x^2 + d_1 x^3$$

   such that $f(x) = f_1(x)$ for all $x \leq \xi$. Express a_1, b_1, c_1, d_1 in terms of $\beta_0, \beta_1, \beta_2, \beta_3, \beta_4$.

   (b) Find a cubic polynomial

   $$f_2(x) = a_2 + b_2 x + c_2 x^2 + d_2 x^3$$

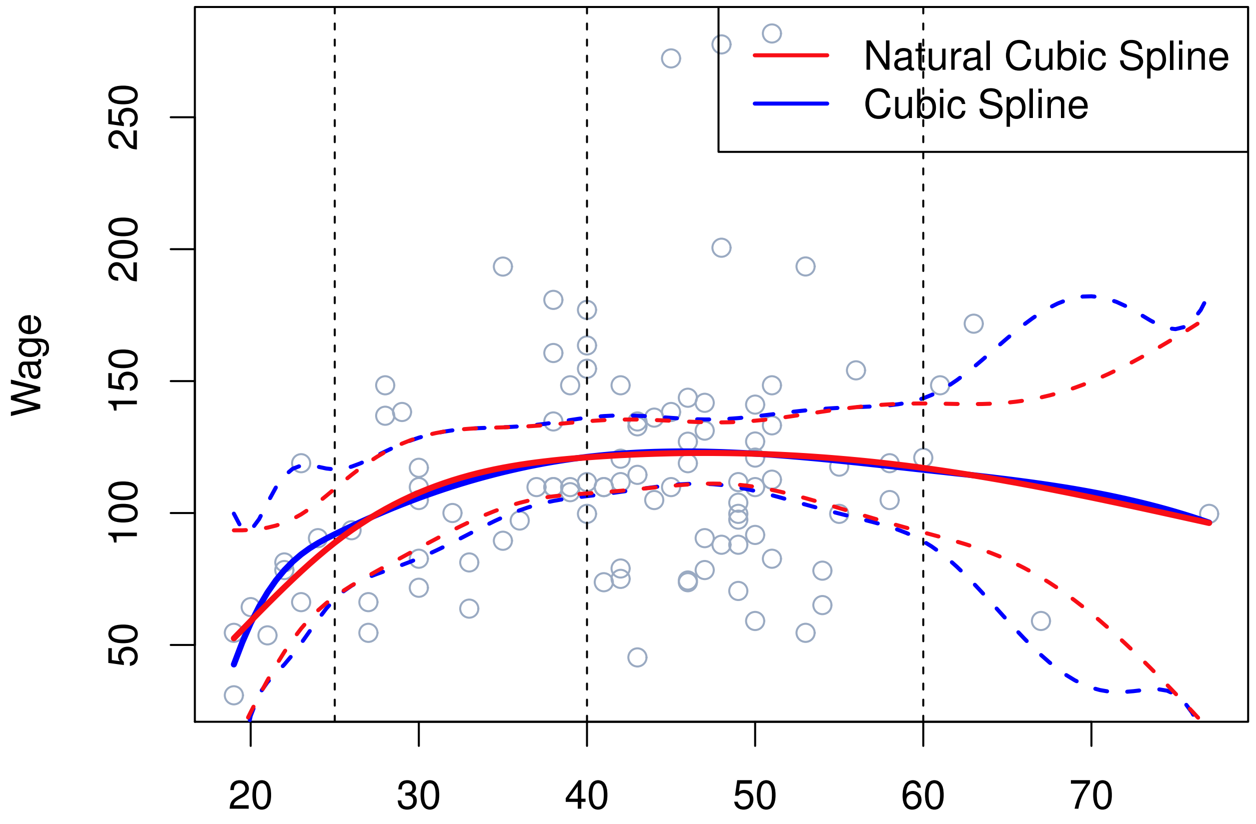
   such that $f(x) = f_2(x)$ for all $x > \xi$. Express a_2, b_2, c_2, d_2 in terms of $\beta_0, \beta_1, \beta_2, \beta_3, \beta_4$. (We would then have established that $f(x)$ is a piecewise polynomial.)

(c) Show that $f_1(\xi) = f_2(\xi)$. That is $f(x)$ is continuous at ξ.

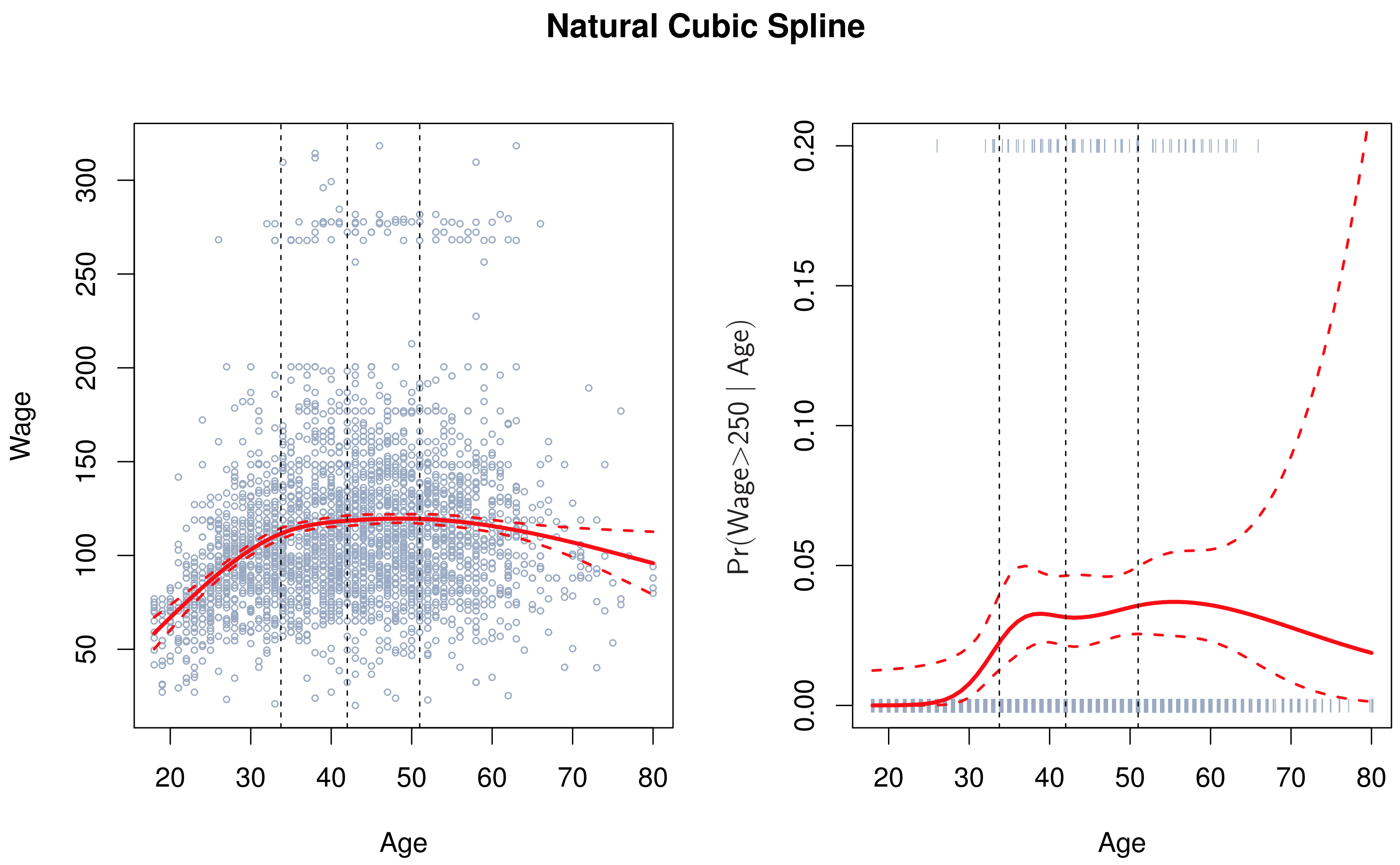
(d) Show that $f_1'(\xi) = f_2'(\xi)$. That is $f'(x)$ is continuous at ξ.

Natural Cubic Splines

A natural cubic spline extrapolates linearly beyond the boundary knots. This adds $4 = 2 \times 2$ extra constraints, and allows us to put more internal knots for the same degrees of freedom as a regular cubic spline.

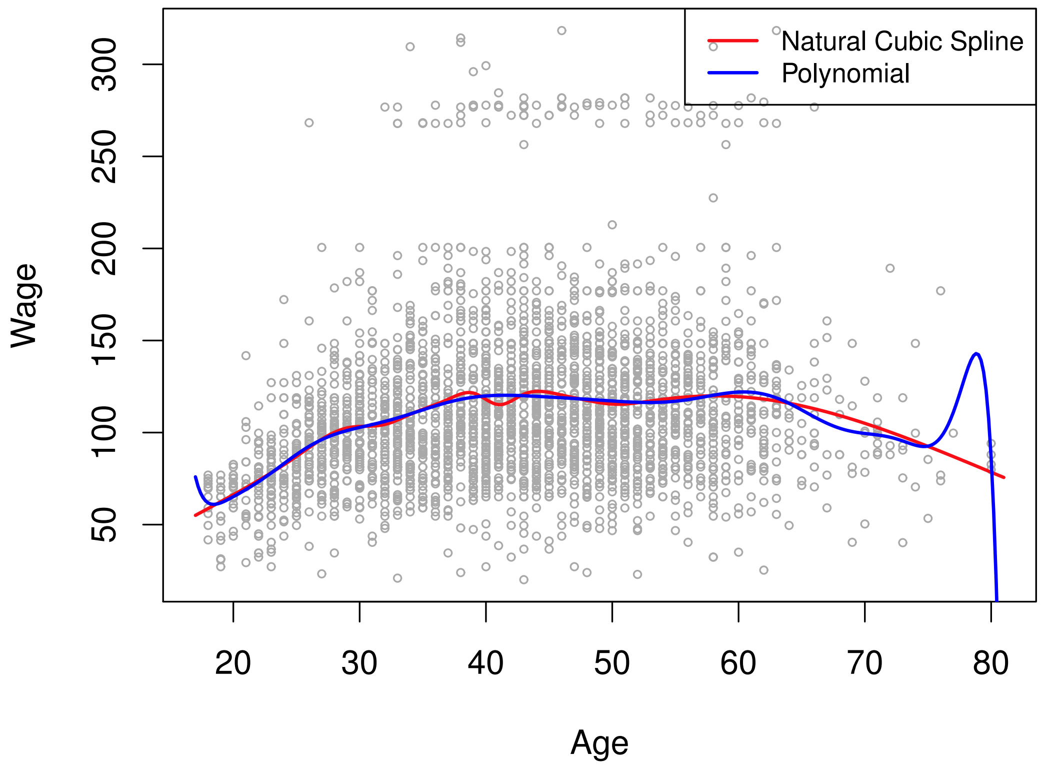


Fitting splines in R is easy: `bs(x, ...)` for any degree splines, and `ns(x, ...)` for natural cubic splines, in package `splines`.



Knot placement

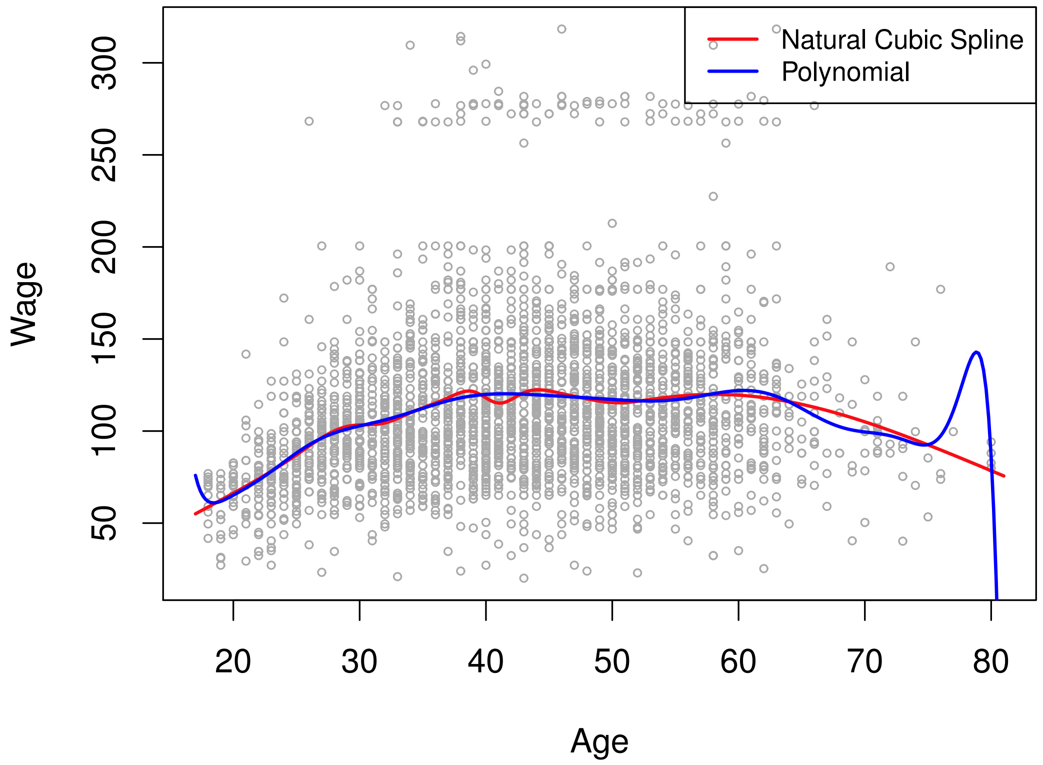
- One strategy is to decide K, the number of knots, and then place them at appropriate quantiles of the observed X.
- A cubic spline with K knots has $K + 4$ parameters or degrees of freedom.
- A natural spline with K knots has K degrees of freedom.



Comparison of a degree-14 polynomial and a natural cubic spline, each with 15df.

Knot placement

- One strategy is to decide K, the number of knots, and then place them at appropriate quantiles of the observed X.
- A cubic spline with K knots has $K + 4$ parameters or degrees of freedom.
- A natural spline with K knots has K degrees of freedom.



Comparison of a degree-14 polynomial and a natural cubic spline, each with 15df.

```
ns(age, df=14)
poly(age, deg=14)
```

Smoothing Splines

Broadly speaking, the second derivative of a function g(t) is a measure of its roughness: it is large in absolute value if g(t) is very wiggly near t, and it is close to zero otherwise. (The second derivative of a straight line is zero; note that a line is perfectly smooth.)

$\int g''(t)^2 dt$ simply a measure of the total change in the function g'(t), over its entire range. If g is very smooth, then g'(t) will be close to constant and $\int g''(t)^2 dt$ will take on a small value. Conversely, if g is jumpy and variable then g"(t) will vary significantly and $\int g''(t)^2 dt$ will take on a large value.

A smoothing spline is simply a natural cubic spline with knots at every unique value of xi. It might seem that a smoothing spline will have far too many degrees of freedom, since a knot at each data point allows a great deal of flexibility. But the tuning parameter λ controls the roughness of the smoothing spline, and hence the **effective degrees of freedom**.

Smoothing Splines

This section is a little bit mathematical

Consider this criterion for fitting a smooth function $g(x)$ to some data:

$$\underset{g \in \mathcal{S}}{\text{minimize}} \sum_{i=1}^{n} (y_i - g(x_i))^2 + \lambda \int g''(t)^2 dt$$

Smoothing Splines

This section is a little bit mathematical

Consider this criterion for fitting a smooth function $g(x)$ to some data:

$$\underset{g \in \mathcal{S}}{\text{minimize}} \sum_{i=1}^{n} (y_i - g(x_i))^2 + \lambda \int g''(t)^2 dt$$

- The first term is RSS, and tries to make $g(x)$ match the data at each x_i.

Smoothing Splines

This section is a little bit mathematical

Consider this criterion for fitting a smooth function $g(x)$ to some data:

$$\underset{g \in \mathcal{S}}{\text{minimize}} \sum_{i=1}^{n} (y_i - g(x_i))^2 + \lambda \int g''(t)^2 dt$$

- The first term is RSS, and tries to make $g(x)$ match the data at each x_i.
- The second term is a *roughness penalty* and controls how wiggly $g(x)$ is. It is modulated by the *tuning parameter* $\lambda \geq 0$.

Smoothing Splines

This section is a little bit mathematical

Consider this criterion for fitting a smooth function $g(x)$ to some data:

$$\underset{g \in \mathcal{S}}{\text{minimize}} \sum_{i=1}^{n} (y_i - g(x_i))^2 + \lambda \int g''(t)^2 dt$$

- The first term is RSS, and tries to make $g(x)$ match the data at each x_i.
- The second term is a *roughness penalty* and controls how wiggly $g(x)$ is. It is modulated by the *tuning parameter* $\lambda \geq 0$.
  - The smaller λ, the more wiggly the function, eventually interpolating y_i when $\lambda = 0$.

Smoothing Splines

This section is a little bit mathematical

Consider this criterion for fitting a smooth function $g(x)$ to some data:

$$\underset{g \in \mathcal{S}}{\text{minimize}} \sum_{i=1}^{n} (y_i - g(x_i))^2 + \lambda \int g''(t)^2 dt$$

- The first term is RSS, and tries to make $g(x)$ match the data at each x_i.
- The second term is a *roughness penalty* and controls how wiggly $g(x)$ is. It is modulated by the *tuning parameter* $\lambda \geq 0$.
  - The smaller λ, the more wiggly the function, eventually interpolating y_i when $\lambda = 0$.
  - As $\lambda \to \infty$, the function $g(x)$ becomes linear.

Smoothing Splines continued

The solution is a natural cubic spline, with a knot at every unique value of x_i. The roughness penalty still controls the roughness via λ.

Smoothing Splines continued

The solution is a natural cubic spline, with a knot at every unique value of x_i. The roughness penalty still controls the roughness via λ.

Some details

- Smoothing splines avoid the knot-selection issue, leaving a single λ to be chosen.

Smoothing Splines continued

The solution is a natural cubic spline, with a knot at every unique value of x_i. The roughness penalty still controls the roughness via λ.

Some details

- Smoothing splines avoid the knot-selection issue, leaving a single λ to be chosen.
- The algorithmic details are too complex to describe here. In R, the function `smooth.spline()` will fit a smoothing spline.

Smoothing Splines continued

The solution is a natural cubic spline, with a knot at every unique value of x_i. The roughness penalty still controls the roughness via λ.

Some details

- Smoothing splines avoid the knot-selection issue, leaving a single λ to be chosen.
- The algorithmic details are too complex to describe here. In R, the function `smooth.spline()` will fit a smoothing spline.
- The vector of n fitted values can be written as $\hat{\mathbf{g}}_\lambda = \mathbf{S}_\lambda \mathbf{y}$, where $\mathbf{S}_\lambda$ is a $n \times n$ matrix (determined by the x_i and λ).
- The *effective degrees of freedom* are given by

$$df_\lambda = \sum_{i=1}^{n} \{\mathbf{S}_\lambda\}_{ii}.$$

Smoothing Splines continued — choosing λ

- We can specify *df* rather than λ!

  In R: `smooth.spline(age, wage, df = 10)`

Smoothing Splines continued — choosing λ

- We can specify *df* rather than λ!

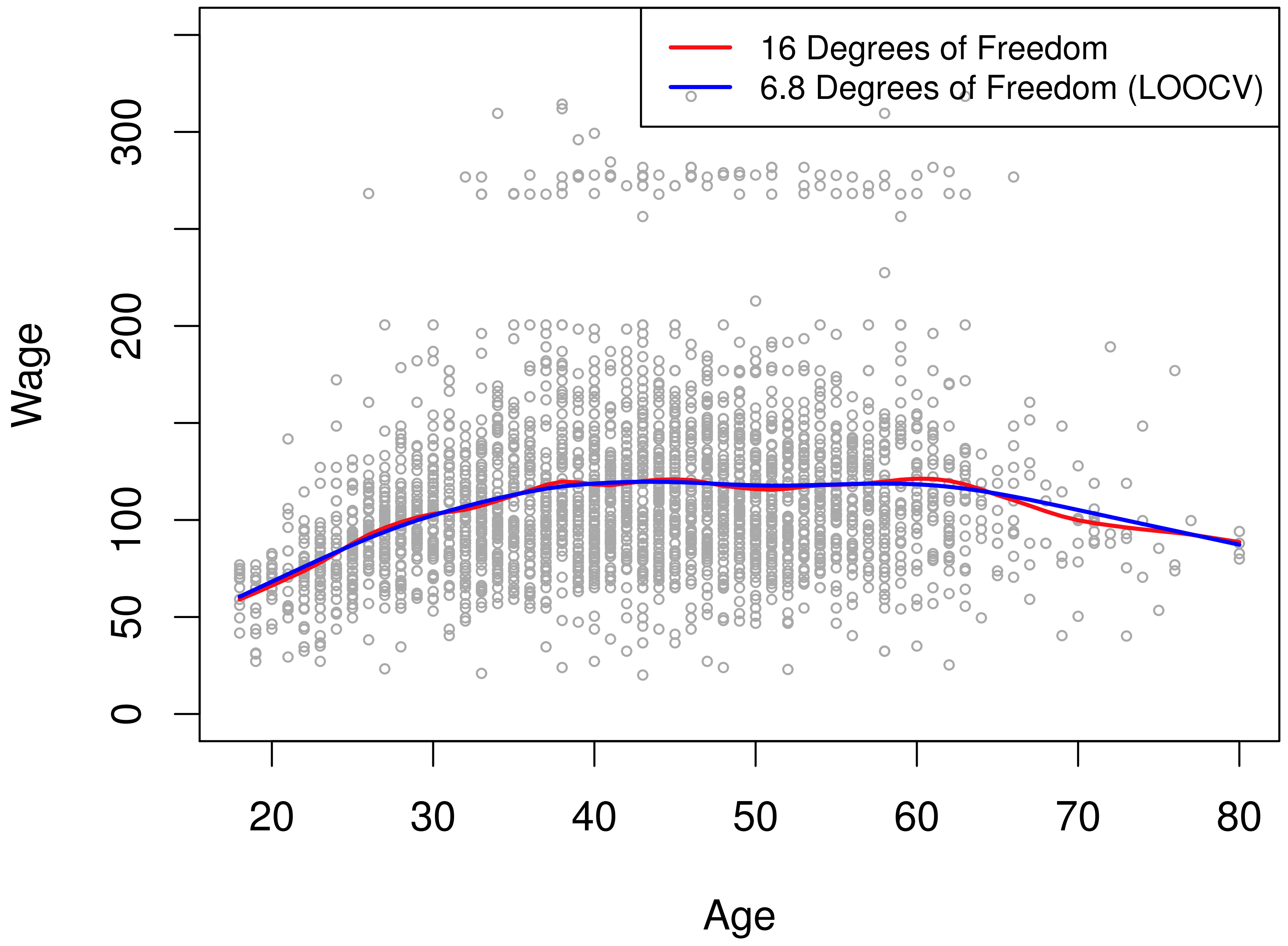
  In R: `smooth.spline(age, wage, df = 10)`

- The leave-one-out (LOO) cross-validated error is given by

$$\text{RSS}_{cv}(\lambda) = \sum_{i=1}^{n}(y_i - \hat{g}_\lambda^{(-i)}(x_i))^2 = \sum_{i=1}^{n}\left[\frac{y_i - \hat{g}_\lambda(x_i)}{1 - \{\mathbf{S}_\lambda\}_{ii}}\right]^2.$$

  In R: `smooth.spline(age, wage)`

Smoothing Spline



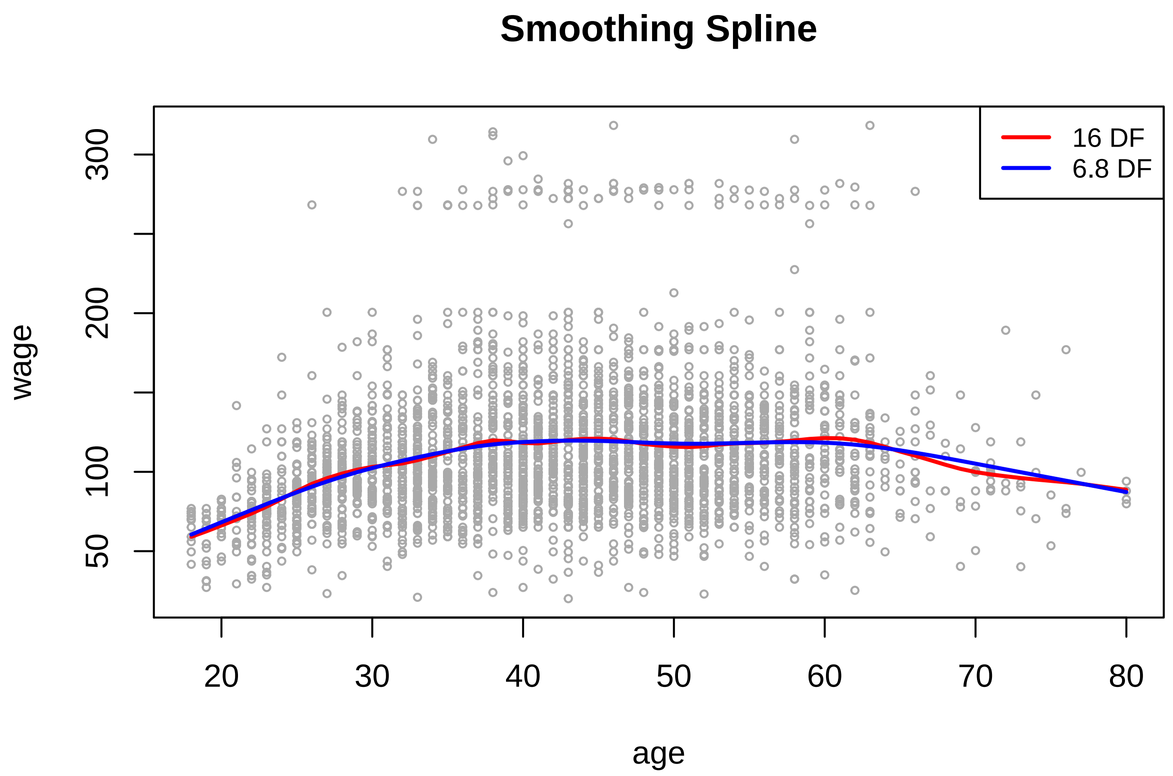
Smoothing Splines in R

In order to fit a smoothing spline, we use the smooth.spline() function. Figure 7.8 was produced with the following code:

```r
par(err=-1)
plot(age, wage, xlim = agelims, cex = 0.5, col = "darkgrey")
title("Smoothing Spline")
fit  <- smooth.spline(age, wage, df = 16)
fit2 <- smooth.spline(age, wage, cv = TRUE)
fit2$df
```

```
## [1] 6.794596
```

```r
lines(fit , col = "red" , lwd = 2)
lines(fit2, col = "blue", lwd = 2)
legend("topright", legend = c("16 DF", "6.8 DF"), col = c("red", "blue"),
lty = 1, lwd = 2, cex = 0.8)
```



Notice that in the first call to smooth.spline(), we specified df=16. The function then determines which value of λ leads to 16 degrees of freedom. In the second call to smooth.spline(), we select the smoothness level by **crossvalidation**; this results in a value of λ that yields 6.8 degrees of freedom.

5. **Exercises, Conceptual**

Suppose that a curve $\hat{g}$ is computed to smoothly fit a set of n points using the following formula

$$\hat{g} = \arg\min_{g}\left(\sum_{i=1}^{n}(y_i - g(x_i))^2 + \lambda \int [g^{(m)}(x)]^2 dx\right),$$

where $g^{(m)}$ represents the mth derivative of g (and $g^{(0)} = g$). Provide example sketches of $\hat{g}$ in each of the following scenarios.

(a) $\lambda = \infty$, $m = 0$.

(b) $\lambda = \infty$, $m = 1$.

(c) $\lambda = \infty$, $m = 2$.

(d) $\lambda = \infty$, $m = 3$.

(e) $\lambda = 0$, $m = 3$.

6. **Exercises, Conceptual**

consider two curves, $\hat{g}_1$ and $\hat{g}_2$, defined by

$$\hat{g}_1 = \arg\min_{g}\left(\sum_{i=1}^{n}(y_i - g(x_i))^2 + \lambda \int [g^{(3)}(x)]^2 dx\right)$$

$$\hat{g}_2 = \arg\min_{g}\left(\sum_{i=1}^{n}(y_i - g(x_i))^2 + \lambda \int [g^{(4)}(x)]^2 dx\right)$$

where $g^{(m)}$ represents the mth derivative of g.

(a) As $\lambda \to \infty$, will $\hat{g}_1$ or $\hat{g}_2$ have the smaller training RSS ?

(b) As $\lambda \to \infty$, will $\hat{g}_1$ or $\hat{g}_2$ have the smaller test RSS ?

(c) For $\lambda = 0$, will $\hat{g}_1$ or $\hat{g}_2$ have the smaller training and test RSS ?

7. **Exercises, Coding**

This question uses the variables "dis" (the weighted mean of distances to five Boston employment centers) and "nox" (nitrogen oxides concentration in parts per 10 million) from the "Boston" data. We will treat "dis" as the predictor and "nox" as the response.

(a) Use the "poly()" function to fit a cubic polynomial regression to predict "nox" using "dis". Report the regression output, and plot the resulting data and polynomial fits.

(b) Plot the polynomial fits for a range of different polynomial degrees (say, from 1 to 10), and report the associated residual sum of squares.

(c) Perform cross-validation or another approach to select the optimal degree for the polynomial, and explain your results.

(d) Use the "bs()" function to fit a regression spline to predict "nox" using "dis". Report the output for the fit using four degrees of freedom. How did you choose the knots ? Plot the resulting fit.

(e) Now fit a regression spline for a range of degrees of freedom, and plot the resulting fits and report the resulting RSS. Describe the results obtained.

(f) Perform cross-validation or another approach in order to select the best degrees of freedom for a regression spline on this data. Describe your results.

8. **Exercises, Coding**

   (a)

Appendix I

polyreg, an Alternative to Neural Networks

r-bloggers: neural networks are essentially polynomial regression

polyreg on github

Appendix II

Reparameterization of Splines

Export fitted regression splines (constructed by 'bs' or 'ns') as piecewise polynomials

Take for instance the following one-knot, degree two, spline:

```
##
## Call:
## lm(formula = wage ~ bs(age, knots = c(42), degree = 2), data = Wage)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -99.356 -24.434  -5.099  15.263 204.754
##
## Coefficients:
```

```
##                                    Estimate Std. Error t value Pr(>|t|)
## (Intercept)                          57.349      3.950  14.518  < 2e-16
## bs(age, knots = c(42), degree = 2)1   59.511      5.786  10.285  < 2e-16
## bs(age, knots = c(42), degree = 2)2   65.722      4.076  16.122  < 2e-16
## bs(age, knots = c(42), degree = 2)3   37.170      9.722   3.823 0.000134
##
## (Intercept)                        ***
## bs(age, knots = c(42), degree = 2)1 ***
## bs(age, knots = c(42), degree = 2)2 ***
## bs(age, knots = c(42), degree = 2)3 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 39.91 on 2996 degrees of freedom
## Multiple R-squared:  0.08631,    Adjusted R-squared:  0.0854
## F-statistic: 94.34 on 3 and 2996 DF,  p-value: < 2.2e-16
```

These coefficients are difficult to interpret.

```
## 2 piecewise polynomials of degree 2 are constructed!
## Use 'summary' to export all of them.
## The first 2 are printed below.
## -5.74e-14 + 4.96 * (x - 18) + 0.0991 * (x - 18) ^ 2
## 61.9 + 0.2 * (x - 42) + 0.0224 * (x - 42) ^ 2
```

```
##               [,1]        [,2]
## [1,] -5.743249e-14 61.91542748
## [2,]  4.959286e+00  0.20033307
## [3,] -9.914485e-02 -0.02240887
```

```
## [1] 18 42 80
```

The function defaults to parametrize piecewise polynomials in shifted form (see ?PiecePoly). You can set shift = FALSE for a non-shifted version.

```
## 2 piecewise polynomials of degree 2 are constructed!
## Use 'summary' to export all of them.
## The first 2 are printed below.
## -121 + 8.53 * x + 0.0991 * x ^ 2
## 14 + 2.08 * x + 0.0224 * x ^ 2
```

```
##               [,1]        [,2]
## [1,] -121.39007747 13.97219046
## [2,]    8.52850050  2.08267822
## [3,]   -0.09914485 -0.02240887
```

You can predict the splines with predict.

```
## [1] TRUE
```

But since there is an intercept in the model, the predicted values would differ from model prediction by the intercept.

```
## [1] TRUE
```