04. Non-linear Models

Table of Contents

Non-linear Models

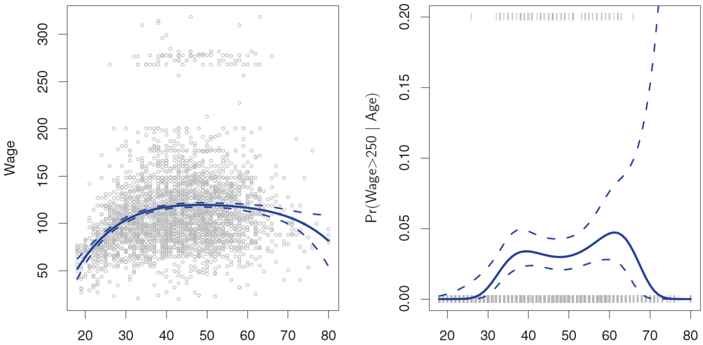
- Linear models are relatively simple and have advantages over other approaches in terms of interpretation and inference.
- The truth is never linear! Or almost never! But, often the linearity assumption is good enough.
- The following non-linear models
  - Polynomials
  - Step functions
  - Splines
  - Local regression
  - Generalized additive models

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

Polynomial Regression

- Polynomial regression extends the linear model by adding extra predictors, obtained by raising each of the original predictors to a power.

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



Polynomial Regression

- We are 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 $\hat{\beta}_l$, we 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

$$\hat{f}(x_0) \pm 2 \cdot \text{se}\left[\hat{f}(x_0)\right].$$

- We either fix the degree d at some reasonably low value, else use cross-validation to choose d.

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

```
## Orthogonal polynomials:
## Each column is a linear orthogonal combination of
## age, age^2, age^3 and age^4
fit <- lm(wage ~ poly(age, 4), data=Wage)
summary(fit)
```

```
## Direct power of age
fit2 <- lm(wage ~ poly(age, 4, raw=T), data=Wage)
summary(fit2)
```

```
fit2a <- lm(wage ~ age + I(age^2) + I(age^3) + I(age^4), Wage)
fit2b <- lm(wage ~ cbind(age, age ^2, age^3, age^4), Wage)
```

```
round(data.frame(fit=coef(fit), fit2=coef(fit2),
                 fit2a=coef(fit2a), fit2b=coef(fit2b)), 5)
```

```r
age.grid <- seq(min(age), max(age))
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)
```

```r
plot(age, wage, xlim=range(age), cex=.5, col="darkgrey")
title("Degree-4 Polynomial", outer=T)
lines(age.grid, preds$fit, lwd=2, col="darkblue")
matlines(age.grid, se.bands, lwd=2, col="darkblue", lty=2)
```

```r
## Orthogonal vs. Non-orthogonal polynomial regression
preds2 <- predict(fit2, newdata=list(age=age.grid), se=TRUE)
data.frame(fit=preds$fit, fit2=preds2$fit)
sum(abs(preds$fit-preds2$fit))
```

```r
## Anova test to find the optimal polynomial degree
fit.1 <- lm(wage ~ age, data=Wage)
fit.2 <- lm(wage ~ poly(age, 2), data=Wage)
fit.3 <- lm(wage ~ poly(age, 3), data=Wage)
fit.4 <- lm(wage ~ poly(age, 4), data=Wage)
fit.5 <- lm(wage ~ poly(age, 5), data=Wage)
g <- anova(fit.1, fit.2, fit.3, fit.4, fit.5)
g
```

```r
## Perform T-test
coef(summary(fit.5))
round(coef(summary(fit.5)), 5)
```

```r
## T-test^2 = F-test
summary(fit.5)$coef[-c(1, 2), 3]
summary(fit.5)$coef[-c(1, 2), 3]^2
g$F[-1]
```

```r
## Covariate effect
fit.1 <- lm(wage ~ education + age , data=Wage)
fit.2 <- lm(wage ~ education + poly(age, 2), data=Wage)
fit.3 <- lm(wage ~ education + poly(age, 3), data=Wage)
fit.4 <- lm(wage ~ education + poly(age, 4), data=Wage)
anova(fit.1, fit.2, fit.3, fit.4)
```

```r
## 10-fold cross-validation to choose the optimal polynomial
set.seed(1111)
N <- 10   ## simulation replications
K <- 10   ## 10-fold CV
```

```r
CVE <- matrix(0, N, 10)
for (k in 1:N) {
    gr <- sample(rep(seq(K), length=nrow(Wage)))
    pred <- matrix(NA, nrow(Wage), 10)
```

```
    for (i in 1:K) {
        tran <- (gr != i)
        test <- (gr == i)
        for (j in 1:10) {
            g <- lm(wage ~ poly(age, j), data=Wage, subset=tran)
            yhat <- predict(g, data.frame(poly(age, j)))
            mse <- (Wage$wage - yhat)^2
            pred[test, j] <- mse[test]
        }
    }
    CVE[k, ] <- apply(pred, 2, mean)
}
RES <- apply(CVE, 2, mean)
RES
```

```
par(mfrow=c(1,2))
matplot(t(CVE), type="l",xlab="Degrees of Polynomials ",
        ylab="Mean Squared Error")
plot(seq(10), RES, type="b", col=2, pch=20, xlab="Degrees of
     Polynomials ", ylab="Mean Squared Error")
```

Polynomial Regression

- Logistic regression can be applied. For example, in the figure we model

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

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

```r
## Logistic regression using binary response
## 1 for wage > 250 and 0 for wage <= 250
fit <- glm(I(wage>250) ~ poly(age, 4), Wage, family="binomial")
```

```r
preds <- predict(fit, newdata=list(age=age.grid), se=T)
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))
```

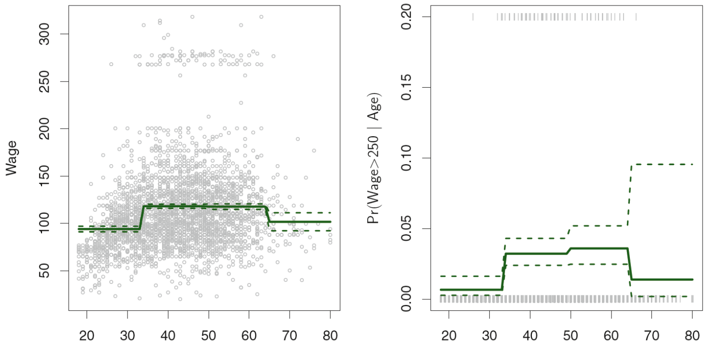
```r
preds2 <- predict(fit, newdata=list(age=age.grid),
                  type="response", se=T)
cbind(pfit, preds2$fit)
```

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

Step Functions

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

$$C_1(X) = I(X < c_1),\ C_2(X) = I(c_1 \leq X < c_2), \ldots$$
$$C_K(X) = I(X \geq c_K)$$



Step Functions

- Need to create a series of dummy variables representing each group.
- Notice that for any value of X,

$$C_0(X) + C_1(X) + \ldots + C_K(X) = 1,$$

  since X must be in exactly one of the $K + 1$ intervals.
- Choice of cutpoints or knots can be problematic. For creating nonlinearities, smoother alternatives such as splines are available.
- Unless there are natural breakpoints in the predictors, piecewise-constant functions can miss some action such as increasing or decreasing trend.

```r
## cut() automatically picked the cut points.
table(cut(age, 4))
fit <- lm(wage ~ cut(age, 4), data=Wage)
fit2 <- glm(I(wage>250) ~ cut(age, 4), data=Wage,
            family="binomial")
```

```r
## The age < 33.5 category is left out
coef(summary(fit))
```

```r
## Fitted values along with confidence bands
age.grid <- seq(min(age), max(age))
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)
```

```r
preds2 <- predict(fit2, newdata=list(age=age.grid), se=T)
pfit <- exp(preds2$fit)/(1 + exp(preds2$fit))
se.bands.logit <- cbind(preds2$fit + 2*preds2$se.fit,
                        preds2$fit - 2*preds2$se.fit)
se.bands2 <- exp(se.bands.logit)/(1 + exp(se.bands.logit))
```

```
par(mfrow=c(1,2), mar=c(4.5 ,4.5 ,1 ,1), oma=c(0, 0, 4, 0))
```

```
plot(age, wage, xlim=range(age), cex=.5, col="darkgrey")
title ("Degree-4 Step Functions", outer=T)
lines(age.grid, preds$fit, lwd=3, col="darkgreen")
matlines(age.grid, se.bands, lwd=2, col="darkgreen", lty=2)
```

```
plot(age , I(wage > 250), xlim=range(age), type="n",
     ylim=c(0, .2))
points(jitter(age), I((wage >250)/5), cex=.5, pch="|",
       col="darkgrey")
lines(age.grid, pfit, lwd=3, col="darkgreen")
matlines(age.grid, se.bands2, lwd=2, col="darkgreen", lty=2)
```

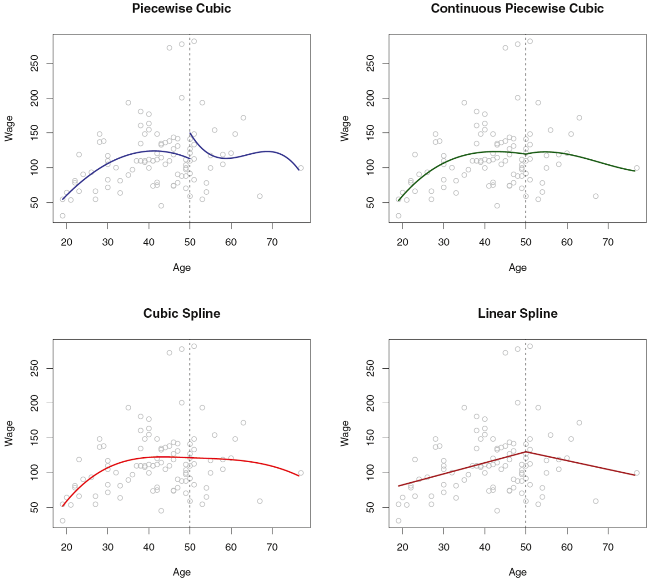
Piecewise Polynomials

- Instead of a single polynomial in X over its whole domain, we can use different polynomials in regions defined by knots.
- For example,

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

- Each of these polynomial functions can be fit using least squares applied to simple functions of the original predictor.
- Using more knots leads to a more flexible piecewise polynomial.
- Better to add constraints to the polynomials, e.g. continuity.
- Splines have the "maximum" amount of continuity.

Piecewise Polynomials



```r
## 200 obs. are randomly generated from 3000 obs.
set.seed(19)
ss <- sample(3000, 200)
nWage <- Wage[ss, ]
age.grid <- seq(min(nWage$age), max(nWage$age))
```

```r
g1 <- lm(wage ~ poly(age, 3), data=nWage, subset=(age < 50))
g2 <- lm(wage ~ poly(age, 3), data=nWage, subset=(age > 50))
```

```r
pred1 <- predict(g1, newdata=list(age=age.grid[age.grid < 50]))
pred2 <- predict(g2, newdata=list(age=age.grid[age.grid >= 50]))
```

```r
par(mfrow = c(1, 2))
plot(nWage[, 2], nWage[, 11], col="darkgrey", xlab="Age",
     ylab="Wage")
title(main = "Piecewise Cubic")
lines(age.grid[age.grid < 50], pred1, lwd=2, col="darkblue")
lines(age.grid[age.grid >= 50], pred2, lwd=2, col="darkblue")
abline(v=50, lty=2)
```

```
## Define the two hockey-stick functions
LHS <- function(x) ifelse(x < 50, 50-x, 0)
RHS <- function(x) ifelse(x < 50, 0, x-50)
```

```
## Fit continuous piecewise polynomials
g3 <- lm(wage ~ poly(LHS(age), 3) + poly(RHS(age), 3), nWage)
pred3 <- predict(g3, newdata=list(age=age.grid))
```

```
plot(nWage[, 2], nWage[, 11], col="darkgrey", xlab="Age",
     ylab="Wage")
title(main="Continuous Piecewise Cubic")
lines(age.grid, pred3, lwd=2, col="darkgreen")
abline(v=50, lty=2)
```

```
summary(g1)
summary(g2)
summary(g3)
```

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

$$y_i = \beta_0 + \beta_1 b_1(x_i) + \beta_2 b_2(x_i) + \ldots + \beta_{K+1} b_{K+1}(x_i) + \epsilon_i$$

where $b_k(\cdot)$ is the basis function.

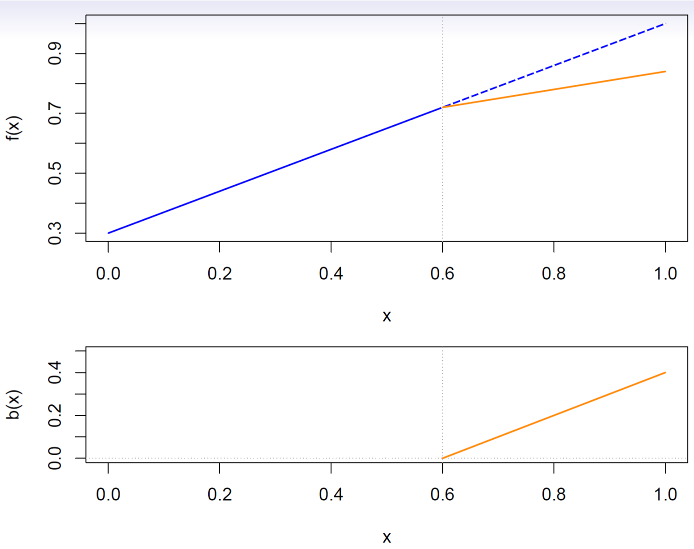
$$\begin{aligned} b_1(x_i) &= x_i \\ b_{k+1}(x_i) &= (x_i - \xi_k)_+ , \end{aligned}$$

for $k = 1, \ldots, K$.

- Here the $(\cdot)_+$ means positive part; i.e.,

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

Linear Splines



Cubic Splines

- Cubic spline with knots at ξ_k, $k = 1, \ldots, K$ is a piecewise cubic polynomial with continuous derivatives up to order 2 at each knot.
- This model with truncated power basis functions is

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

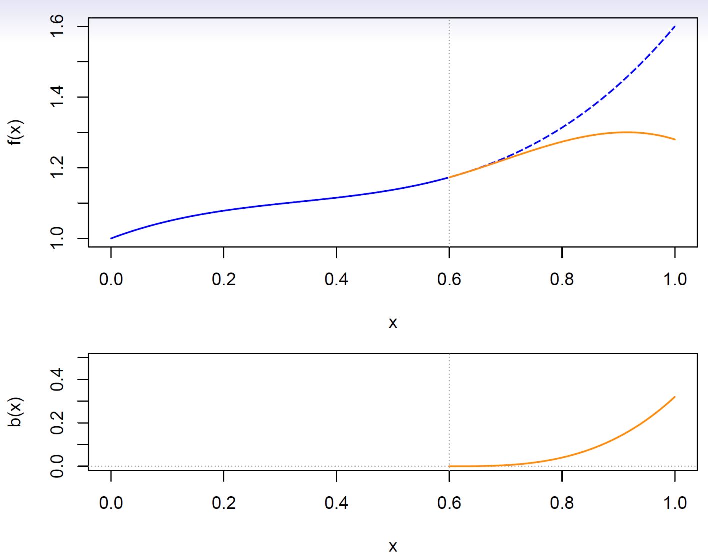
where $b_k(\cdot)$ is the basis function;

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

for $k = 1, \ldots, K$, and

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

Cubic Splines



```r
## Truncated power basis functions
d <- 3
knots <- 50
x1 <- outer(nWage$age, 1:d, "^")
x2 <- outer(nWage$age, knots, ">") *
      outer(nWage$age, knots, "-")^d
x <- cbind(x1, x2)
g4 <- lm(wage ~ x, data=nWage)
```

```r
nx1 <- outer(age.grid, 1:d, "^")
nx2 <- outer(age.grid, knots, ">") *
       outer(age.grid, knots, "-")^d
nx <- cbind(nx1, nx2)
pred4 <- predict(g4, newdata=list(x=nx))
```

```r
par(mfrow=c(1,2))
plot(nWage[, 2], nWage[, 11], col="darkgrey", xlab="Age",
     ylab="Wage")
title(main = "Cubic Spline")
lines(age.grid, pred4, lwd=2, col="red")
abline(v=50, lty = 2)
```

```r
library(splines)
g5 <- lm(wage ~ bs(age, knots=50), data=nWage)
pred5 <- predict(g5, newdata=list(age=age.grid))
```

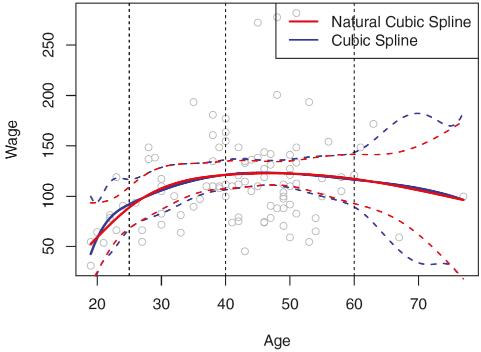
```r
plot(nWage[, 2], nWage[, 11], col="darkgrey", xlab="Age",
     ylab="Wage")
title(main="Cubic Spline")
lines(age.grid, pred5, lwd=2, col="red")
abline(v=50, lty=2)
```

```r
## Linear spline
g6 <- lm(wage ~ bs(age, knots=50, degree=1), data=nWage)
pred6 <- predict(g6, newdata=list(age=age.grid))
```

```r
plot(nWage[, 2], nWage[, 11], col="darkgrey", xlab="Age",
     ylab="Wage")
title(main="Linear Spline")
lines(age.grid, pred6, lwd=2, col="darkred")
abline(v=50, lty=2)
```

Natural Cubic Splines

- Splines can have high variance at the outer range of the predictors. i.e., when X is very small or very large.
- A natural spline is a regression spline with additional boundary constraints: the natural function is required to be linear at the boundary. So, natural splines generally produce more stable estimates at the boundaries.



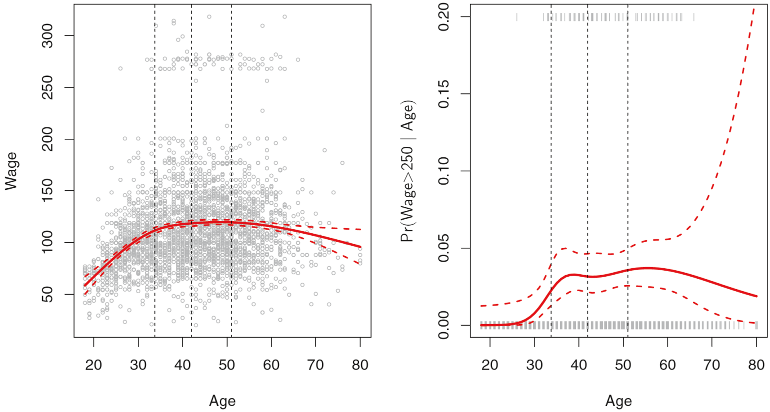
```r
## Cubic Spline
fit <- lm(wage ~ bs(age, knots=c(25 ,40 ,60)), data=nWage)
pred <- predict(fit, newdata=list(age=age.grid), se=T)
```

```r
## Natural Spline
fit2 <- lm(wage ~ ns(age, knots=c(25 ,40 ,60)), data=nWage)
pred2 <- predict(fit2, newdata=list(age=age.grid), se=T)
```

```r
plot(nWage[, 2], nWage[, 11], col="darkgrey", xlab="Age",
     ylab="Wage")
lines(age.grid, pred$fit, lwd=2, col=4)
lines(age.grid, pred$fit + 2*pred$se, lty="dashed", col=4)
lines(age.grid, pred$fit - 2*pred$se, lty="dashed", col=4)
lines(age.grid, pred2$fit, lwd=2, col=2)
lines(age.grid, pred2$fit + 2*pred2$se, lty="dashed", col=2)
lines(age.grid, pred2$fit - 2*pred2$se, lty="dashed", col=2)
abline(v=c(25, 40, 60), lty=2)
legend("topright", c("Natural Cubic Spline", "Cubic Spline"),
       lty=1, lwd=2, col=c(2, 4))
```

Natural Cubic Splines

- We fit a natural cubic spline with three knots, where the knot locations were chosen automatically as the 25th, 50th, and 75th percentiles.



```r
## Use a complete Wage data
age <- Wage$age
wage <- Wage$wage
age.grid <- seq(min(age), max(age))
```

```r
g1 <- lm(wage ~ ns(age, df=4), data=Wage)
pred1 <- predict(g1, newdata=list(age=age.grid), se=T)
se.bands1 <- cbind(pred1$fit + 2*pred1$se.fit,
                   pred1$fit - 2*pred1$se.fit)
```

```r
g2 <- glm(I(wage > 250) ~ ns(age, df=4), data=Wage,
          family="binomial")
pred2 <- predict(g2, newdata=list(age=age.grid), se=T)
pfit <- exp(pred2$fit)/(1 + exp(pred2$fit))
se.bands.logit <- cbind(pred2$fit + 2*pred2$se.fit,
                        pred2$fit - 2*pred2$se.fit)
se.bands2 <- exp(se.bands.logit)/(1 + exp(se.bands.logit))
```

```
par(mfrow=c(1,2), mar=c(4.5 ,4.5 ,1 ,1), oma=c(0, 0, 4, 0))
```

```
plot(age, wage, cex=.5, col="darkgrey", xlab="Age",
     ylab="Wage")
title("Natural Cubic Spline", outer=T)
lines(age.grid, pred1$fit, lwd=3, col=2)
matlines(age.grid, se.bands1, lwd=2, col=2, lty=2)
ncs <- ns(age, df=4)
attr(ncs, "knots")
abline(v=attr(ncs, "knots"), lty=3)
```

```
plot(age, I(wage > 250), type ="n", ylim=c(0, .2), xlab="Age",
     ylab="Pr(Wage>250 | Age)")
points(jitter(age), I((wage >250)/5), cex=.5, pch ="|",
       col="darkgrey")
lines(age.grid, pfit, lwd=3, col=2)
matlines(age.grid, se.bands2, lwd=2, col=2, lty=2)
abline(v=attr(ncs, "knots"), lty=3)
```

Natural Cubic Splines

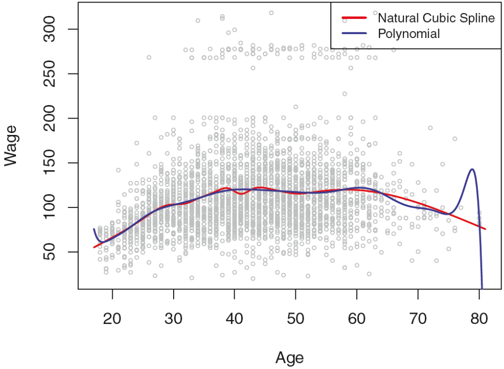
- When we fit a spline, where should we place the knots?
  - Place more knots where the function might vary most rapidly.
  - Place fewer knots where it seems more stable.
- How many knots should we use, or equivalently how many degrees of freedom should our spline contain?
  - Use a cross-validation.
- Please note that 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.

```
set.seed(1111)
CVE <- matrix(0, 20, 10)
for (k in 1:20) {
    gr <- sample(rep(seq(10), length=nrow(Wage)))
    pred <- matrix(NA, nrow(Wage), 10)
    for (i in 1:10) {
        tran <- (gr != i)
        test <- (gr == i)
        for (j in 1:10) {
            nsx <- ns(age, df=j)
            g <- lm(wage ~ nsx, data=Wage, subset=tran)
            mse <- (Wage$wage - predict(g, nsx))^2
            pred[test, j] <- mse[test]
        }
     }
     CVE[k, ] <- apply(pred, 2, mean)
}
```

```
RES <- apply(CVE, 2, mean)
plot(seq(10), RES, type="b", col=2, pch=20, xlab="Degrees of
     Freedom of Natural Spline", ylab="Mean Squared Error")
```

Comparison to Polynomial Regression

- Regression splines often give superior results to polynomial regression.
- The extra flexibility in the polynomial produces undesirable results at the boundaries, while the natural cubic spline still provides a reasonable fit to the data.



```r
g1 <- lm(wage ~ ns(age, df=15), data=Wage)
g2 <- lm(wage ~ poly(age, 15), data=Wage)
pred1 <- predict(g1, newdata=list(age=age.grid), se=T)
pred2 <- predict(g2, newdata=list(age=age.grid), se=T)
```

```r
plot(age, wage, cex=.5, col="darkgrey", xlab="Age",
     ylab="Wage")
lines(age.grid, pred1$fit, lwd=2, col=2)
lines(age.grid, pred1$fit + 2*pred1$se, lty="dashed", col=2)
lines(age.grid, pred1$fit - 2*pred1$se, lty="dashed", col=2)
```

```r
lines(age.grid, pred2$fit, lwd=2, col=4)
lines(age.grid, pred2$fit + 2*pred2$se, lty="dashed", col=4)
lines(age.grid, pred2$fit - 2*pred2$se, lty="dashed", col=4)
legend("topleft", c("Natural Cubic Spline", "Polynomial"),
        lty=1, lwd=2, col=c(2, 4))
```

Smoothing Splines

- We want to find a function $g(x)$ that makes RSS small, but that is also smooth.
- A function $g(x)$ that minimizes below is a smoothing spline.

$$\sum_{i=1}^{n} (y_i - g(x_i))^2 + \lambda \int g^{''}(t)^2 dt$$

  where λ is a nonnegative tuning parameter.
- The second term is a roughness penalty and controls how wiggly $g(x)$ is. It is modulated by $\lambda \geq 0$.
  - 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

- 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 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

    $$\hat{\boldsymbol{g}}_\lambda = \boldsymbol{S}_\lambda y,$$

    where $\boldsymbol{S}_\lambda$ is a $n \times n$ matrix determined by the x_i and λ.
  - The effective degrees of freedom are given by

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

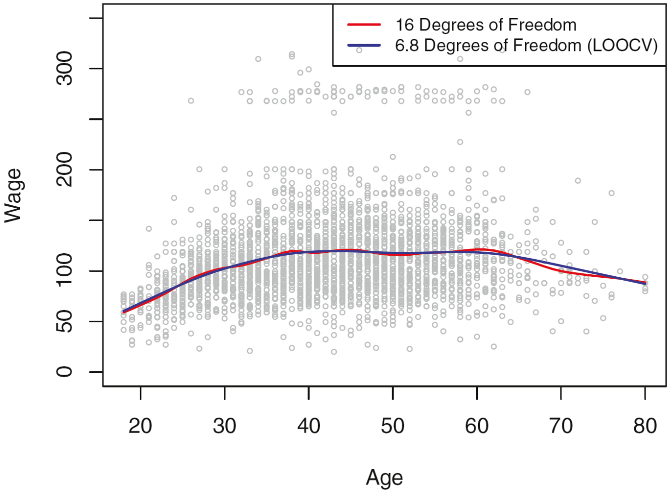
Smoothing Splines

- In fitting a smoothing spline, we do not need to select the number of knots, but need to choose the value of λ.
- One possible solution is cross-validation.
- The leave-one-out cross-validation error (LOOCV) can be computed very efficiently for smoothing splines

$$\mathsf{LOOCV}_\lambda = \sum_{i=1}^{n} \left(y_i - \hat{g}_\lambda^{[-i]}(x_i) \right)^2 = \sum_{i=1}^{n} \left[\frac{y_i - \hat{g}_\lambda(x_i)}{1 - \{\boldsymbol{S}_\lambda\}_{ii}} \right]^2,$$

  where $\hat{g}_\lambda^{[-i]}(x_i)$ is the fitted value without the ith observation.
- We can specify the effective degrees of freedom `df` rather than λ in R.

Smoothing Splines



```r
library(ISLR)
library(splines)
data(Wage)
age <- Wage$age
wage <- Wage$wage
age.grid <- seq(min(age), max(age))
```

```r
fit <- smooth.spline(age, wage, df=16)
fit2 <- smooth.spline(age, wage, cv=TRUE)
fit2$df
fit3 <- lm(wage ~ ns(age, df=7), data=Wage)
pred3 <- predict(fit3, newdata=list(age=age.grid))
```

```r
plot(age, wage, cex=.5, col = "darkgrey")
title("Smoothing Spline vs Natural Spline")
lines(fit, col="red", lwd =2)
lines(fit2, col="blue", lwd =2)
lines(age.grid, pred3, col="green", lwd=2)
legend("topright", legend=c("SS (DF=16)", "SS (DF=6.8)",
       "NS (DF=7)"), col=c("red","blue","green"), lty=1, lwd=2)
```

```r
set.seed(1234)
N <- 10          ## Simulation replications
K <- 10          ## 10-fold CV
df <- seq(2, 20) ## Degrees of freedom
CVE <- matrix(0, N, length(df))
```

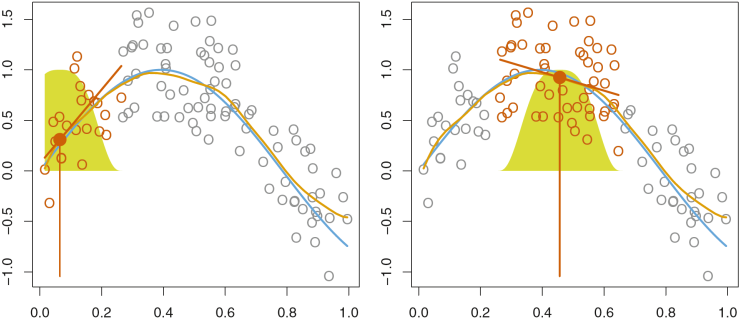
```r
for (k in 1:N) {
    gr <- sample(rep(seq(K), length=nrow(Wage)))
    pred <- matrix(NA, nrow(Wage), length(df))
    for (i in 1:K) {
        tran <- (gr != i)
        test <- (gr == i)
        for (j in 1:length(df)) {
            fit <- smooth.spline(age[tran], wage[tran], df=df[j])
            mse <- (wage-predict(fit, age)$y)^2
            pred[test, j] <- mse[test]
        }
    }
CVE[k, ] <- apply(pred, 2, mean)
}
RES <- apply(CVE, 2, mean)
```

```r
par(mfrow=c(1,2))
matplot(t(CVE), type="b", col=2, lty=2, pch=20, ylab="CV errors",
        xlab="Degrees of freedom")
plot(df, RES, type="b", col=2, pch=20, ylab="averaged CV errors",
     xlab="Degrees of freedom")
abline(v=df[which.min(RES)], col="grey", lty=4)
```

```r
set.seed(1357)
MSE1 <- matrix(0, 100, 2)
for (i in 1:100) {
    tran <- sample(nrow(Wage), size=floor(nrow(Wage)*2/3))
    test <- setdiff(1:nrow(Wage), tran)
    g1 <- smooth.spline(age[tran], wage[tran], df=7)
    g2 <- lm(wage ~ ns(age, df=7), data=Wage, subset=tran)
    mse1 <- (wage-predict(g1, age)$y)[test]^2
    mse2 <- (wage-predict(g2, Wage))[test]^2
    MSE1[i,] <- c(mean(mse1), mean(mse2))
}
apply(MSE1, 2, mean)
```

Local Regression

- Local regression computes the fit at a target point x_0 using only the regression nearby training observations.
- With a sliding weight function, we fit separate linear fits over the range of x by weighted least squares.



Local Regression

Algorithm 7.1 *Local Regression At* $X = x_0$

1. Gather the fraction $s = k/n$ of training points whose x_i are closest to x_0.

2. Assign a weight $K_{i0} = K(x_i, x_0)$ to each point in this neighborhood, so that the point furthest from x_0 has weight zero, and the closest has the highest weight. All but these k nearest neighbors get weight zero.

3. Fit a *weighted least squares regression* of the y_i on the x_i using the aforementioned weights, by finding $\hat{\beta}_0$ and $\hat{\beta}_1$ that minimize

$$\sum_{i=1}^{n} K_{i0}(y_i - \beta_0 - \beta_1 x_i)^2. \tag{7.14}$$

4. The fitted value at x_0 is given by $\hat{f}(x_0) = \hat{\beta}_0 + \hat{\beta}_1 x_0$.

```r
data(Wage)
age <- Wage$age
wage <- Wage$wage
age.grid <- seq(min(age), max(age))
```

```r
fit1 <- loess(wage ~ age, span=.2, data=Wage)
fit2 <- loess(wage ~ age, span=.7, data=Wage)
```

```r
plot(age, wage, cex =.5, col = "darkgrey")
title("Local Linear Regression")
lines(age.grid, predict(fit1, data.frame(age=age.grid)),
      col="red", lwd=2)
lines(age.grid, predict(fit2, data.frame(age=age.grid)),
      col="blue", lwd=2)
legend("topright", legend = c("Span = 0.2", "Span = 0.7"),
       col=c("red", "blue"), lty=1, lwd=2)
```

```r
## Degrees of freedom
c(fit1$enp, fit2$enp)
```

```r
set.seed(1357)
MSE2 <- matrix(0, 100, 2)
for (i in 1:100) {
    tran <- sample(nrow(Wage), size=floor(nrow(Wage)*2/3))
    test <- setdiff(1:nrow(Wage), tran)
    g1 <- loess(wage ~ age, span=.2, data=Wage)
    g2 <- loess(wage ~ age, span=.7, data=Wage)
    mse1 <- (wage-predict(g1, Wage))[test]^2
    mse2 <- (wage-predict(g2, Wage))[test]^2
    MSE2[i,] <- c(mean(mse1), mean(mse2))
}
MSE <- cbind(MSE1, MSE2)
apply(MSE, 2, mean)
apply(MSE, 2, sd)
```

```r
boxplot(MSE, boxwex=0.5, col=4:7, ylim=c(1200, 2000), names=
        c("Smoothing Spline", "Natural Cubic", "Local (S=0.2)",
        "Local (S=0.7)"), ylab="Mean Squared Errors")
```

Generalized Additive Models

- We predict Y on the basis of several predictors $x_1, \ldots, x_p$.
- Generalized additive models (GAMs) allows non-linear functions of each of the variables, while maintaining additivity.

$$y_i = \beta_0 + f_1(x_{i1}) + f_2(x_{i2}) + \cdots + f_p(x_{ip}) + \epsilon_i$$

- It is called an additive model because we calculate a separate f_j for each x_j, and then add together all of their contributions.
- The non-linear fits can potentially make more accurate predictions for the response Y.
- GAMs provide a useful compromise between linear and fully nonparametric models.

```
gam1 <- lm(wage ~ ns(year, 4) + ns(age, 5) + education, data=Wage)
summary(gam1)
```

```
library(gam)
```

```
## s() : smoothing spline
gam <- gam(wage ~ s(year, 4)+s(age, 5)+education, data=Wage)
par(mfrow =c(1,3))
plot(gam, se=TRUE, col="blue", scale=70)
plot.Gam(gam1, se = TRUE, col = "red")
```

```
## Significance test
gam.m1 <- gam(wage ~ s(age, 5) + education, data=Wage)
gam.m2 <- gam(wage ~ year + s(age, 5) + education, data=Wage)
anova(gam.m1, gam.m2, gam, test = "F")
summary(gam)
```

```
set.seed(1357)
MSE3 <- matrix(0, 100, 3)
```

```
for (i in 1:100) {
    tran <- sample(nrow(Wage), size=floor(nrow(Wage)*2/3))
    test <- setdiff(1:nrow(Wage), tran)
    g1 <- gam(wage ~ s(age, 5) + education, data=Wage,
              subset=tran)
    g2 <- gam(wage ~ year + s(age, 5) + education, data=Wage,
              subset=tran)
    g3 <- gam(wage ~ s(year, 4) + s(age, 5) + education,
              data=Wage, subset=tran)
    mse1 <- (wage - predict(g1, Wage))[test]^2
    mse2 <- (wage - predict(g2, Wage))[test]^2
    mse3 <- (wage - predict(g3, Wage))[test]^2
    MSE3[i,] <- c(mean(mse1), mean(mse2), mean(mse3))
}
apply(MSE3, 2, mean)
```

```r
## lo(): local regression
gam.lo <- gam(wage ~ s(year, df=4) + lo(age, span=0.7) +
              education, data=Wage)
```

```r
par(mfrow =c(1,3))
plot(gam.lo, se=TRUE, col="blue", scale=70)
```

```r
table(Wage$education, I(wage > 250))
```

```r
gam.lr.s <- gam(I(wage > 250) ~ year + s(age, df=5) + education,
                family="binomial", data=Wage,
                subset=(education != "1. < HS Grad"))
```

```r
par(mfrow = c(1, 3))
plot(gam.lr.s, se=T, col="green", scale=10)
```