Linear Regression with Nonlinear Effects

Chun Li, PhD

Division of Biostatistics

Department of Population and Public Health Sciences

University of Southern California

June 12-16, 2023

Linear Regression

When the outcome is a continuous variable, we may use **linear regression**. The model is

$$Y = \beta_0 + \beta^T \boldsymbol{X} + \epsilon = \beta_0 + \beta_1 X_1 + \cdots + \beta_p X_p + \epsilon,$$

where the residual ϵ has mean 0 and variance σ^2. Here $\beta = (\beta_1, \ldots, \beta_p)$ is a vector of coefficients, and $\beta^T \boldsymbol{X} = \beta_1 X_1 + \cdots + \beta_p X_p$.

When we have independent observations $\{(y_i, \boldsymbol{x}_i) : i = 1, \ldots, n\}$, the model is often fit with **least squares**, which is to identify $\hat{\beta}_0$ and $\hat{\beta}$ such that they minimize $\sum_{i=1}^{n}(y_i - \beta_0 - \beta^T \boldsymbol{x}_i)^2$. An implicit assumption here is that all observations have similar residual variances.

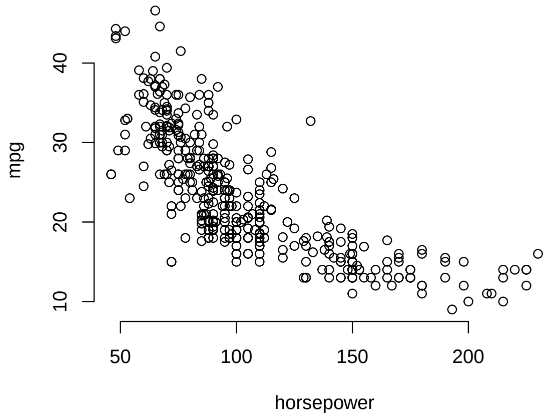
In linear regression, we allocate one **degree of freedom** (DF) to each predictor variable to model its linear effect.

We now go beyond linearity and model the **nonlinear effect** of a predictor.

Nonlinear Effects

An example of nonlinear effects.

```r
par(mar=c(4,4,1,1))
with(ISLR::Auto, plot(horsepower, mpg, bty='n'))
```



Options:

- Add a quadratic term
  (or a quadratic term + a cubic term)
- Polynomial regression with $d \geq 4$ (x)
- Step function (piecewise constant) (x)
- Natural splines (or smooth splines) ($\checkmark$)
- Local regression ($\checkmark$)

Polynomial Regression

A simple approach to modeling nonlinear effect of a variable X_1 is to add a *quadratic term*, X_1^2, to the model (2 DFs for X_1), as in:

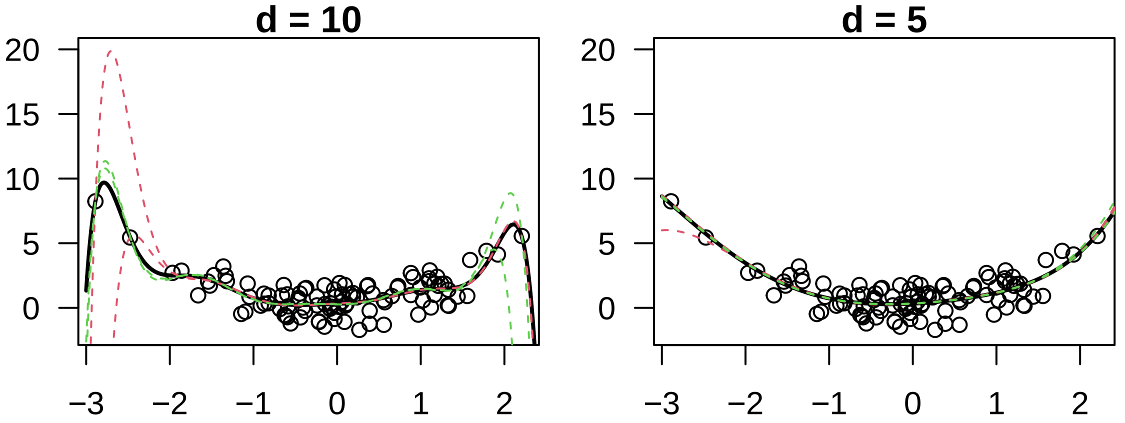
$$Y = \beta_0 + \beta^T \boldsymbol{X} + \epsilon = \beta_0 + \beta_{11} X_1 + \beta_{12} X_1^2 + \beta_2 X_2 + \cdots + \beta_p X_p + \epsilon,$$

Occasionally, a *cubic term*, X_1^3, may be added, with 3 DFs for X_1.

But polynomial regression with a higher degree (≥ 4) is generally not recommended, because

- A high-degree polynomial is difficult to interpret.
- A polynomial with K degrees has the same DFs as a natural spline with $K + 1$ knots, but the latter has much better flexibility and interpretation.
- Polynomials allow high leverage data to have too much impact on the result.
- Data at one end could influence the shape of the fitted curve on the other end, because a polynomial is a "global" function.

We now demonstrate the last two points. We first simulate some data ($n = 100$), then fit polynomial regression with $d = 10$ and $d = 5$ (black). We remove an observation that has one of the most extreme x (two lowest x and two highest x), and refit the models (red and green).



Step Functions

A step function transforms a quantitative predictor into a categorical variable. For example, "age group" is sometimes used as a predictor variable instead of age itself. Using a step function on a predictor variable results in a *piecewise constant* model.

- It can lead to distortion and loss of information. For example, when using age group in decades as a predictor, we implicitly assume that ages 41 and 49 (8 years apart) have the same effect, while ages 49 and 51 (2 years apart) have different effects.
- By assuming a constant effect over an interval we may miss a trend inside the interval.
- Too few categories may over-simplify the effect of the variable. Too many categories uses many DFs.
- A step function with K categories has the same DFs as a natural spline with K knots, but the latter has much better flexibility and interpretation.

Categorization of a variable may be useful for reporting results, but often not helpful for analysis.

Splines

A **spline** is a *piecewise polynomial* function, with constraints at the connecting **knots** to ensure the function is smooth.

There are often 3 constraints at every knot t:

- The function has same value at t,
- Its first derivative has same value at t,
- Its second derivative has same value at t.

The rationale of these constraints is shown in Figure 5.2 of *Elements of Statistical Learning*, 2nd Edition.

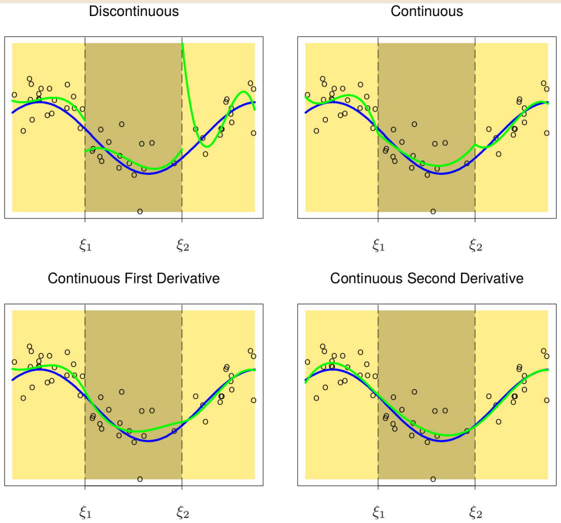


FIGURE 5.2. *A series of piecewise-cubic polynomials, with increasing orders of continuity.*

Cubic Splines

Cubic splines are splines that are piecewise cubic functions.

- A cubic spline with K knots has $K+1$ intervals.
- A cubic spline with K knots has $K+3$ DFs attributable to the variable.
  [Each interval has 4 parameters (because a cubic function has 4 coefficients). There are 3 constraints at every knot. So, there are $4(K+1)-3K=K+4$ DFs, which include one DF for the intercept. Therefore, $K+3$ DFs are attributable to the variable.]
- One set of $K+3$ *basis functions* is:

$$x, x^2, x^3, (x-t_1)_+^3, \cdots, (x-t_K)_+^3,$$

  where $t_1 < \ldots < t_K$ are the K knots, and $(x-t)_+^3 = (x-t)^3$ when $x \geq t$ and 0 when $x < t$. The function $(x-t)_+^3$ is called *truncated power basis function* at t.
- B-splines provide another set of basis functions.

Natural Splines / Restricted Cubic Splines

In cubic splines, the internal intervals have constraints on both sides, but the two end intervals have constraints on only one side, leaving them too flexible. This is addressed by natural splines.

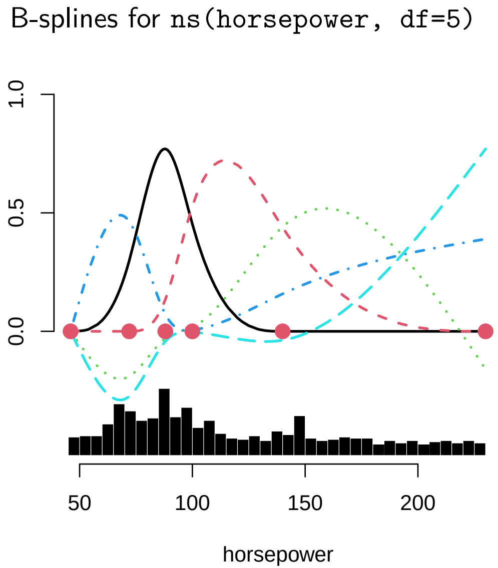
Natural splines (also called **restricted cubic splines** or RCS) are cubic splines but with linear functions (instead of cubic functions) for the leftmost and rightmost intervals.

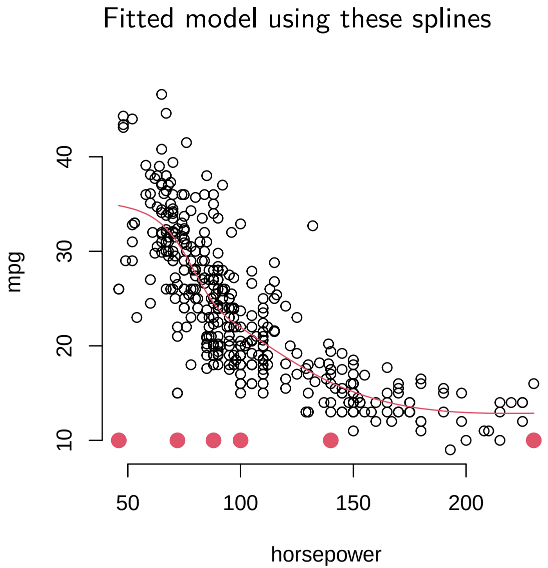
- A natural spline with K knots has $K-1$ DFs attributable to the variable.
  [A natural spline has 2 fewer parameters for each end interval than a cubic spline with the same number of knots.]
- One set of $K-1$ basis functions is: $N_1(x) = x$, and

$$N_{j+1}(x) = (x - t_j)_+^3 - \frac{t_K - t_j}{t_K - t_{K-1}}(x - t_{K-1})_+^3 + \frac{t_{K-1} - t_j}{t_K - t_{K-1}}(x - t_K)_+^3,$$

  for $j = 1, \ldots, K-2$, where $t_1 < \ldots < t_K$ are the K knots. `rms::rcs()` uses a scaled version of this set as basis functions.
- `splines::ns()` uses B-splines to generate a different set of basis functions.

Example Basis Functions





Number of Knots and Positions of Knots

The **number of knots** depends on how many DFs you can, and want to, “spend” on a variable.

- The total number of DFs you can “spend” depends on the sample size and the number of predictor variables.
- For a variable of interest with a clear nonlinear effect, we may want to spend a few DFs on it by modeling its effect with natural splines.
- For a covariate you just want to adjust for, you may or may not model its nonlinear effect.
- The number of knots can be treated as a hyperparameter, which may be selected through cross-validation.

Positions of knots:

- Ideally, knots should be placed at where the function may change rapidly.
- Sometimes, they are chosen at certain default quantiles.
- It is recommended that the smallest knot and the largest knots are put close to or at the ends, because otherwise there might be a visible linear portion at the ends.

ns() and rcs() in R

Suppose we want to spend 5 DFs on variable X. Then there will be 6 knots.

In `ns()` from the splines R package:

- We can specify DF: e.g., `ns(x, df=5)`. The knots include 2 *boundary knots* fixed at the very ends by default (i.e., min(x) and max(x)), and *internal knots* chosen at evenly spaced quantiles (e.g., 20th, 40th, 60th, 80th percentiles in this example).
- We can also specify the positions of the knots explicitly: e.g.,
  `ns(x, knots=c(60,80,150,180))` # leave boundary knots at default
  `ns(x, knots=c(60,80,150,180), Boundary.knots=c(20,270))`

In `rcs()` from the rms R package:

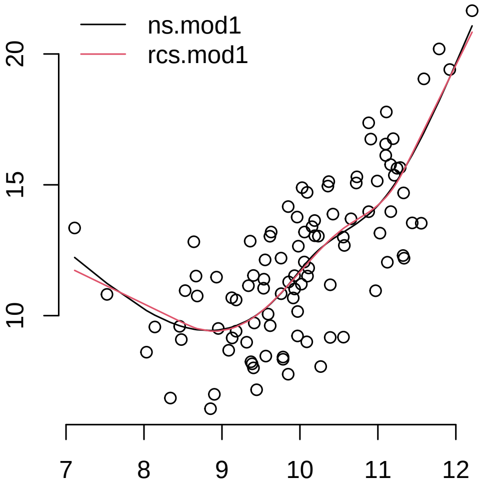
- We can specify the number of knots: e.g. `rcs(x, parms=6)`. The knots are chosen to be the quantiles at 0.05 and 0.95, and other quantiles evenly spaced in between (e.g., 5th, 23th, 41th, 59th, 77th, 95th quantiles in this example).
- We can also specify the knots explicitly: e.g. `rcs(x,parms=c(20,60,80,150,180,270))`

ns() and rcs() in R (cont'd)

```r
## Simulate a dataset with x and y
set.seed(20); n = 100
x = rnorm(n) + 10; y = 10 + (x-9)^2 + rnorm(n,0,2)
par(mar=c(2,2,1,1))
plot(x, y, bty='n')

## Fit spline models using ns() and rcs()
library(splines); library(rms)
ns.mod1 = lm(y ~ ns(x, df=5))
rcs.mod1 = lm(y ~ rcs(x, parms =
          quantile(x, c(.05, .1, .3, .7, .9, .95))))

## Draw the fitted spline functions
idx = order(x)
points(x[idx], fitted(ns.mod1)[idx], type='l', col=1)
points(x[idx], fitted(rcs.mod1)[idx], type='l', col=2)
legend('topleft', col=1:2, lty=1, bty='n',
       legend=c("ns.mod1", "rcs.mod1"))
```



The models are different because they use different knots.

ns() and rcs() in R (cont'd)

Once the knots are the same, ns() and rcs() give the same model.

```
## internal knots from ns(x, df=5)
K1 = with(attributes(ns(x, df=5)), knots)
## boundary knots from ns(x, df=5)
K2 = with(attributes(ns(x, df=5)), Boundary.knots)

## Fit a rcs() model with knots obtained above
rcs.mod2 = lm(y ~ rcs(x, parms=c(K1,K2)))
all.equal(predict(ns.mod1), predict(rcs.mod2))
```

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

```
## Fit a rcs() model with knots explicitly specified
rcs.mod3 = lm(y ~ rcs(x, parms=
      c(range(x), quantile(x, c(.2, .4, .6, .8)))))
all.equal(predict(ns.mod1), predict(rcs.mod3))
```

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

```
rcs.mod4 = ols(y ~ rcs(x, parms=
      c(range(x), quantile(x, c(.2, .4, .6, .8)))))
all.equal(predict(ns.mod1), predict(rcs.mod4))
```

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

But they can have different coefficients.

```
summary(ns.mod1)$coef
```

```
##                   Estimate Std. Error   t value     Pr(>|t|)
## (Intercept)     12.2236594   1.526001 8.0102549 3.038494e-12
## ns(x, df = 5)1 -1.5380952   1.503016 -1.0233394 3.087738e-01
## ns(x, df = 5)2  0.8478543   1.836584 0.4616475 6.454007e-01
## ns(x, df = 5)3  2.6454132   1.316994 2.0086752 4.744040e-02
## ns(x, df = 5)4  2.5691541   3.542401 0.7252580 4.700955e-01
## ns(x, df = 5)5 11.0401089   1.548482 7.1296356 2.041734e-10
```

```
summary(rcs.mod2)$coef
```

```
##                                  Estimate Std. Error     t value
## (Intercept)                     29.537554  11.403157  2.59029611 0
## rcs(x, parms = c(K1, K2))x      -2.435050   1.405077 -1.73303679 0
## rcs(x, parms = c(K1, K2))x'      7.122827   4.074437  1.74817462 0
## rcs(x, parms = c(K1, K2))x''    -1.947134  89.126924 -0.02184675 0
## rcs(x, parms = c(K1, K2))x''' -118.320587 243.258608 -0.48639836 0
## rcs(x, parms = c(K1, K2))x''''  176.117066 208.297990  0.84550536 0
```

This is because they use different basis functions for their splines. In other words, they have different parameterizations for their splines.

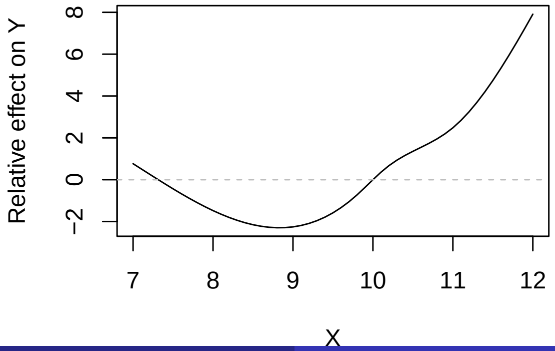
Interpretation of Models with Splines

It is often difficult to interpret the spline coefficients because each coefficient is for a single basis function.

Variables with splines should be evaluated as a whole instead of over individual coefficients.

Evaluation of the effect of a variable: Pick a reference value x_0. Compare the effect at any other x value with that at x_0. For the example on the right, we set $x_0 = 10$.

```
x0 = data.frame(x=10)
xgrid = data.frame(x=seq(7, 12, 0.1))
cc = predict(ns.mod1, newdata=xgrid) -
     predict(ns.mod1, newdata=x0)
par(mar=c(4,4,1,1))
plot(xgrid$x, cc, type='l', xlab='X',
     ylab='Relative effect on Y')
abline(h=0, col='grey', lty=2)
```



Confidence Intervals

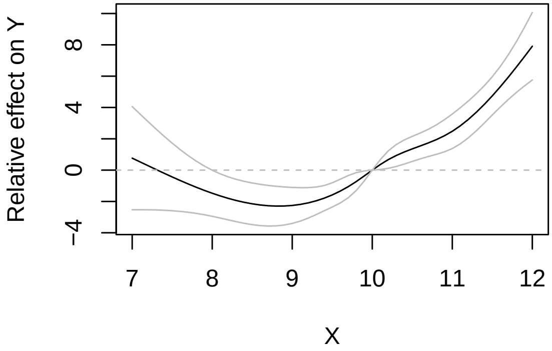
To add confidence intervals, we can use `rcs()` and fit the model with `ols()`. Then we can use `summary()` on the model to extract information for the confidence bands.

```r
rcs.mod4 = ols(y ~ rcs(x, parms=
      c(range(x), quantile(x, c(.2, .4, .6, .8)))))

x0 = data.frame(x=10)
xgrid = data.frame(x=seq(7, 12, 0.1))
cc2 = predict(rcs.mod4, newdata=xgrid) -
      predict(rcs.mod4, newdata=x0)

CI = matrix(0, nrow(xgrid), 2)
for(i in 1:nrow(xgrid)) {
  CI[i,] = summary(rcs.mod4,
                   x=c(10,10,xgrid$x[i]))[6:7]
}
```

```r
par(mar=c(4,4,1,1))
plot(xgrid$x, cc2, type='l', ylim=range(CI),
     xlab='X', ylab='Relative effect on Y')
abline(h=0, col='grey', lty=2)
points(xgrid$x, CI[,1], type='l', col='grey')
points(xgrid$x, CI[,2], type='l', col='grey')
```



Testing for Significant Nonlinearity

To test if the extra DFs are significant, we can use a likelihood ratio test, in which we compare

- the model with splines (the full model)
- the model with only the linear term (the null model)

We can use `anova()` for this test.

```
## the null model
mod0 = lm(y ~ x)

anova(mod0, ns.mod1)
```

```
## Analysis of Variance Table
##
## Model 1: y ~ x
## Model 2: y ~ ns(x, df = 5)
##   Res.Df    RSS Df Sum of Sq      F    Pr(>F)
## 1     98 501.51
## 2     94 370.74  4    130.77 8.2892 9.025e-06 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

```
anova(mod0, rcs.mod1)
```

```
## Analysis of Variance Table
##
## Model 1: y ~ x
## Model 2: y ~ rcs(x, parms = quantile(x, c(0.05, 0.1, 0.3, 0.7, 0.9,
##   Res.Df    RSS Df Sum of Sq     F  Pr(>F)
## 1     98 501.51
## 2     94 374.43  4    127.08 7.976 1.4e-05 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

AIC for Selection of DF

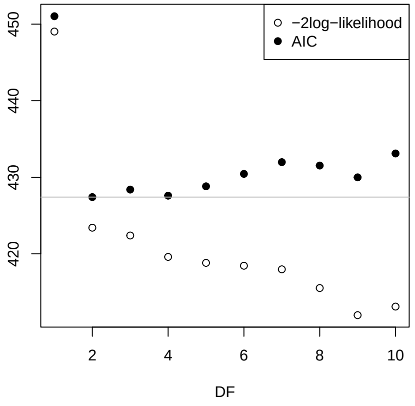
We can determine how many DFs are the most adequate by using AIC (Akaike's Information Criterion).

AIC = -2 x log-likelyhood + 2 x #par

```r
ns.aic = NULL
for(df in 1:10) {
  ns.aic[df] = AIC(lm(y ~ ns(x, df=df)))
}
which.min(ns.aic)
```

```
## [1] 2
```

```r
plot(1:10, ns.aic)
abline(h=ns.aic[which.min(ns.aic)], col='grey')
```



For this data, the minimum AIC occurs when *df* = 2. Thus we choose the spline model with 2 DFs.

Note: The AIC for *df* = 4 is slightly higher than that for *df* = 2. If it were slightly lower than that for *df* = 2, we would have still chosen 2 DFs because it is a more parsimonious model.

Other Options: Smoothing splines

Smoothing splines start with a very different motivation. Consider all smooth functions $g(x)$ such that $g''(x)$ exists. We optimize the following:

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

- A high $|g''(t)|$ reflects a quick change of $g'(t)$, which happens when $g(t)$ is bumpy. So $\int g''(t)^2 dt$ is a way to measure the overall level of bumpiness/roughness of $g(t)$.
- The roughness of $g(t)$ is penalized by multiplying it with λ, which is then added to the sum of squared residuals.
- It can be shown that the best fit $\hat{g}(x)$ is a natural spline with all x_i being knots. So, **a smoothing spline is a regularized natural spline**.
- Smoothing splines are a version of generalized ridge regression.

Effective Degrees of Freedom

It is difficult to specify the penalty λ in smoothing splines. Fortunately for every λ, there is a corresponding **effective degrees of freedom**. The higher effective DF the rougher the function is. The effective DF can be any number in $(1, n_x]$, where n_x is the number of unique x values.

In R, we can use `smooth.spline(x, y, df=)` to fit smoothing spline models.

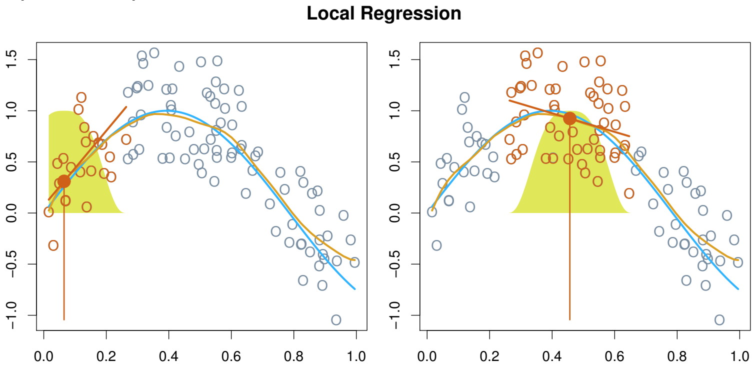
```
require(ISLR)
sms.df4 = with(Auto, smooth.spline(horsepower, mpg, df=4))
```

Alternatively, we can specify the **smoothing parameter** `spar=`, which is a value in $(0, 1]$. The lower `spar` the rougher the function is.

```
sms.spar.5 = with(Auto, smooth.spline(horsepower, mpg, spar=.5))
```

Other Options: Local Regression

Local regression is moving-window weighted regression. For every x_0 in the range of x, we fit a local weighted regression model to obtain the fitted value $\hat{g}(x_0)$. We then move on to the next value and repeat the process.



From *Introduction to Statistical Learning with R*, Figure 7.9.

Other Options: Local Regression (cont'd)

Need to specify:

- **Span**: Fraction of data used for every local regression model.
- **Regression model**: piecewise constant, linear, or quadratic.
- **Kernel**: It determines how data within the span are weighted as a function of relative distance from x_0.

In `loess()`, the default is local *quadratic* model using 75% neighboring data fitted with least squares (i.e., no additional weighting). This often yields a very smooth function.

In `lo()` from the gam R package, the default is local *linear* model using 50% neighboring data.

Generalized Additive Models

Generalized additive models (GAMs) have the form

$$y = \beta_0 + f_1(x_1) + \cdots + f_p(x_p) + \epsilon,$$

where $f_1(), \ldots, f_p()$ can be different functions with different levels of smoothness. If $f_k(x_k) = x_k$, only linear effect is modeled on variable x_k.

- One can plan on the number of DFs spent on each predictor.
- Can be fit using backfitting (or LS when all the functions are explicit).
- Flexible modeling of the effects of individual predictors. The functions can be global or local or piecewise. A function can also be over two or three predictor variables.

Backfitting is an iterative algorithm for fitting additive models. For example, suppose our model is $y = \beta_0 + f_1(x_1) + f_2(x_2) + f_3(x_3) + \epsilon$. Given the current estimates $\hat{\beta}_0$, $\hat{f}_1$, and $\hat{f}_2$, we calculate partial residuals $r_i = y_i - \hat{\beta}_0 - \hat{f}_1(x_i) - \hat{f}_2(x_i)$ and then fit r_i to $f_3(x_i)$ to obtain a new estimate $\hat{f}_3$. We then repeat this process to estimate another component in the model. Repeat several cycles until convergence.

gam() in R

In `gam()` from the gam R package, we can use:

- `s(x)` for a smoothing spline for x,
- `lo(x)` for a loess fit for x,
- Other basis generators such as `ns()`, `bs()`, and `poly()`,
- Traditional model terms such as `x` (linear effect if `x` is quantitative, or categorical effect if `x` is qualitative), `I(x>10)`, and interaction term `x1*x2`, etc.

Here are two example GAM models:

```
library(gam)
library(ISLR)
gam.m3 = gam(wage ~ s(year,4) + s(age,5) + education, data=Wage)
gam.m4 = gam(wage ~ ns(year,4) + lo(age,span=0.4) + education, data=Wage)
```

Interaction Terms in Linear Regression

Consider **an interaction model** between sex and smoking,

$$E(Y) = \beta_0 + \beta_1 \cdot \text{sex} + \beta_2 \cdot \text{smoking} + \beta_3 \cdot \text{sex} \cdot \text{smoking}$$
$$= \begin{cases} \beta_0, & \text{female non-smoker;} \\ \beta_0 + \beta_1, & \text{male non-smoker;} \\ \beta_0 \qquad + \beta_2, & \text{female smoker;} \\ \beta_0 + \beta_1 + \beta_2 + \beta_3, & \text{male smoker.} \end{cases}$$

- β_0 is the <u>mean outcome</u> for the "baseline" group of female non-smokers.
- β_1 is the <u>effect</u> of sex among non-smokers. (The effect of sex among smokers is $\beta_1 + \beta_3$.)
- β_2 is the <u>effect</u> of smoking among females. (The effect of smoking among males is $\beta_2 + \beta_3$.)
- β_3 is a <u>difference of effects</u>:

$$\begin{aligned} \beta_3 &= \text{effect of sex among smokers} - \text{effect of sex among non-smokers} \\ &= \text{effect of smoking among males} - \text{effect of smoking among females.} \end{aligned}$$

Interaction Terms in Linear Regression (cont'd)

When there is an interaction term between X_1 and X_2,

- β_1 and β_2 do not reflect the overall effects of X_1 and X_2, That is, they are not "main effects".
- β_0, β_1, and β_3 are not comparable because they are different concepts and are on different scales:
  - β_0 is mean outcome.
  - β_1 and β_2 are effects (i.e., difference of mean outcomes).
  - β_3 is the difference of two effects (i.e., difference of two differences).
- It is meaningless to make statements like $\beta_1 = \beta_3$ even though they may happen to have the same value.
- It is meaningless to state that "the interaction effect is stronger than the main effect".

Interaction Terms in Linear Regression (cont'd)

Now consider an interaction model between adult age (a continuous variable with values way above 0) and smoking,

$$\begin{aligned} E(Y) &= \beta_0 + \beta_1 \cdot \text{age} + \beta_2 \cdot \text{smoking} + \beta_3 \cdot \text{age} \cdot \text{smoking} \\ &= \begin{cases} \beta_0 + \beta_1 \cdot \text{age}, & \text{non-smoker;} \\ (\beta_0 + \beta_2) + (\beta_1 + \beta_3) \cdot \text{age}, & \text{smoker.} \end{cases} \end{aligned}$$

- β_0 is the mean outcome for non-smokers at 0 year old (**nonexistent!**).
- β_1 is the effect of one year older vs. current age among non-smokers. (The effect of one year older vs. current age among smokers is $\beta_1 + \beta_3$.)
- β_2 is the effect of smoking among 0 year olds (**nonexistent!**) (The effect of smoking among x year olds is $\beta_2 + \beta_3 x$.)
- β_3 is a difference of effects:

$$\begin{aligned} \beta_3 &= \text{effect of one-year-older among smokers} - \text{effect of one-year-older among non-smokers} \\ &= \text{effect of smoking among } x+1 \text{ year olds} - \text{effect of smoking among } x \text{ year olds.} \end{aligned}$$