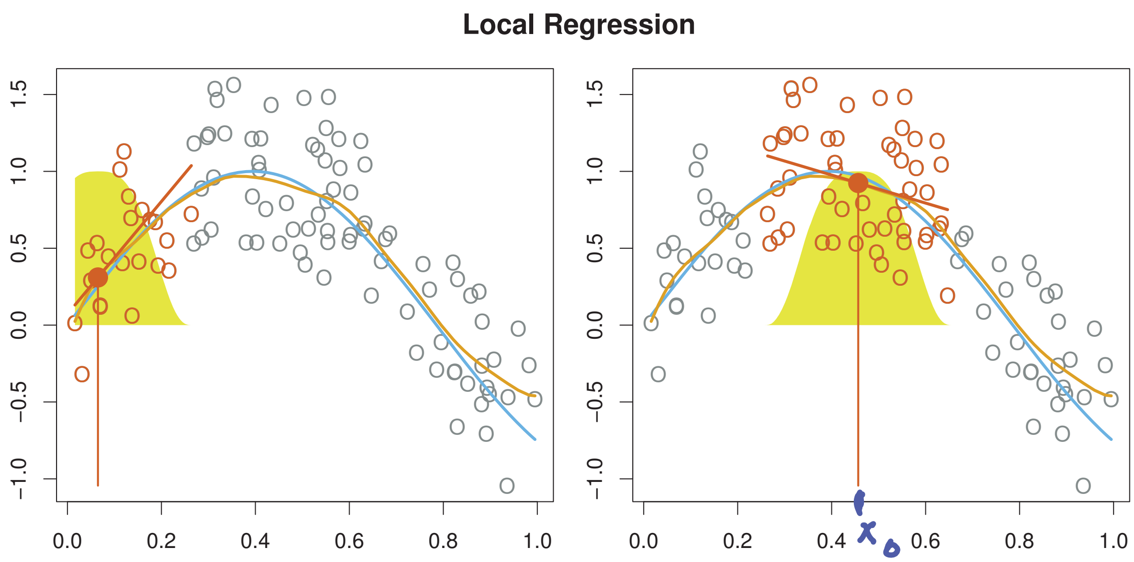
Local Regression

Local regression involves computing the fit at a target point x_0 using only the nearby training observations.



Local regression illustrated on some simulated data, where the blue curve represents $f(x)$ from which the data were generated, and the light orange curve corresponds to the local regression estimate $\hat{f}(x)$. The orange colored points are local to the target point x_0, represented by the orange vertical line. The yellow bell-shape superimposed on the plot indicates weights assigned to each point, decreasing to zero with distance from the target point. The fit $\hat{f}(x_0)$ at x_0 is obtained by fitting a weighted linear regression (orange line segment), and using the fitted value at x_0 (orange solid dot) as the estimate $\hat{f}(x_0)$.

Local Regression at $X = x_0$:

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

   span

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.

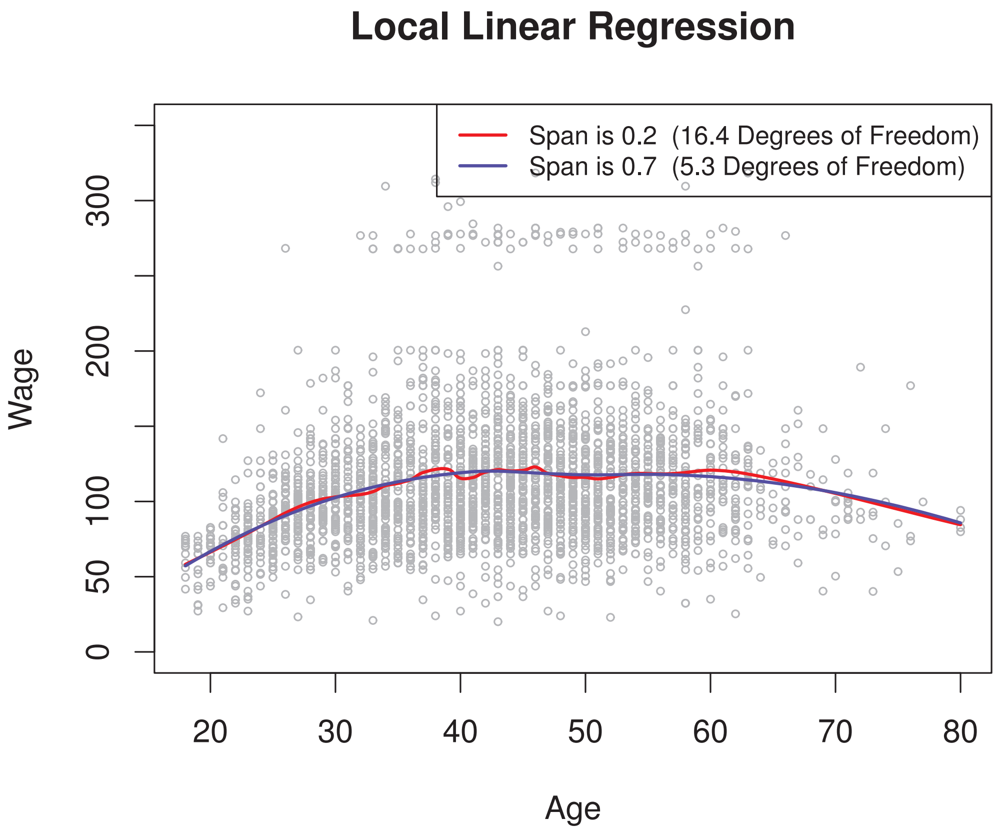
3. Fit a weighted least squares regression of the y_i on the x_i using the aforementioned weights, by finding $\hat{\beta}_0$ and $\hat{\beta}_1$ that minimize

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

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

The span s is the tuning parameter which controls the flexibility of the non-linear fit. The smaller the value of s, the more local and wiggly will be our fit; alternatively a very large value of s will lead to a global fit to the data using all of the training observations.

We can use cross-validation to choose s, or we can specify it directly.



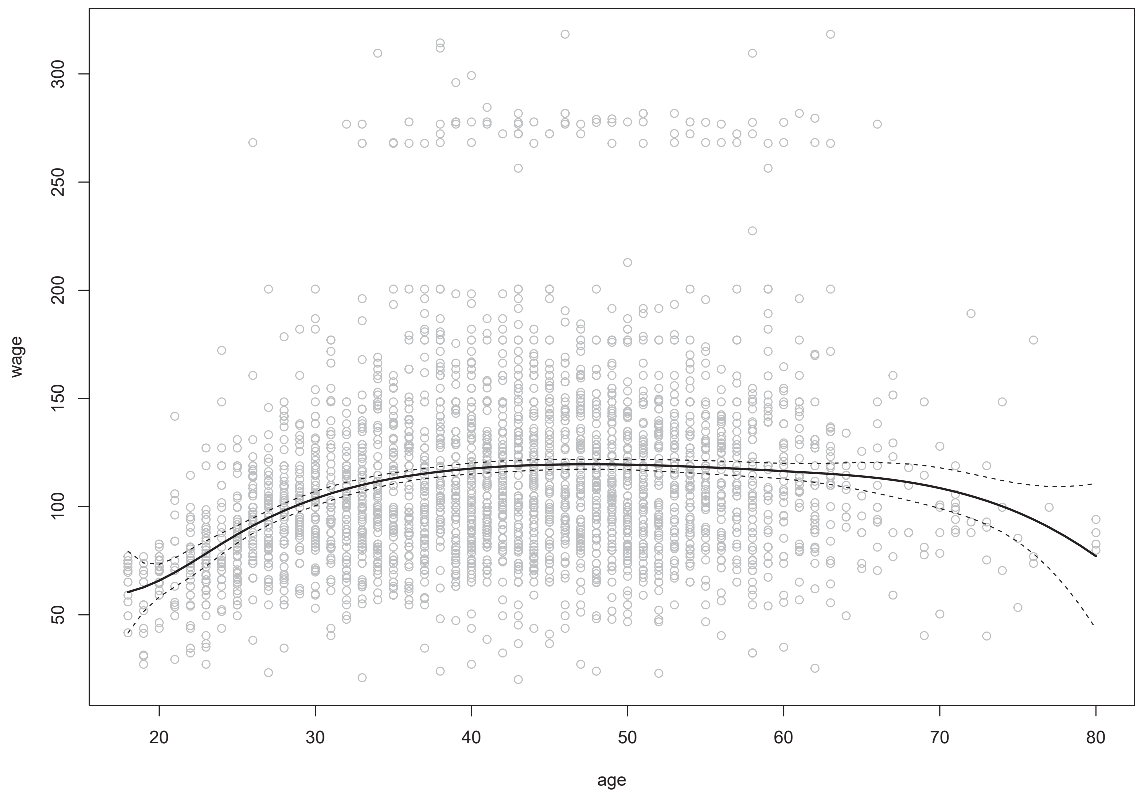
Local linear fits to the Wage data. The span specifies the fraction of the data used to compute the fit at each target point.

Performing Splines, Smoothing Splines, and Local Regression in R

In order to fit regression splines in R, we use the splines library. 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:

```
> library(splines)
> 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")
```



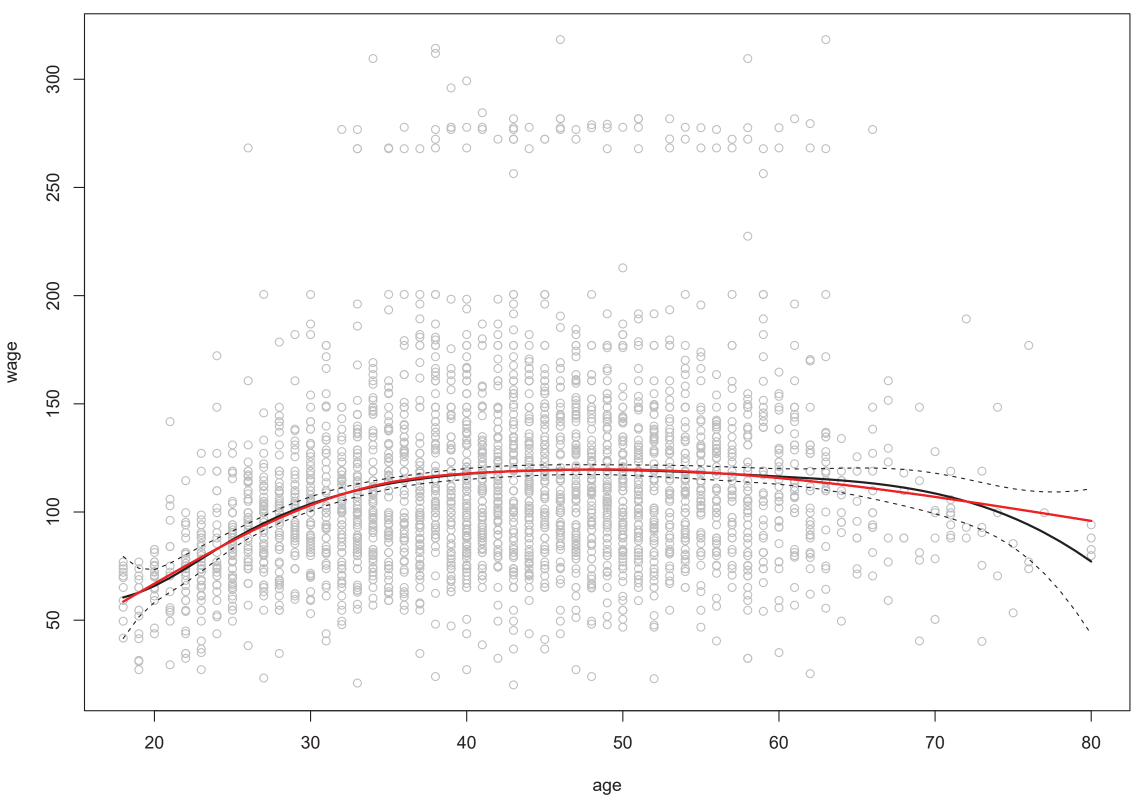
Here we have prespecified knots at ages 25, 40, and 60. This produces a spline with six basis functions. We could also use the df option to produce a spline with knots at uniform quantiles of the data.

```
> 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.

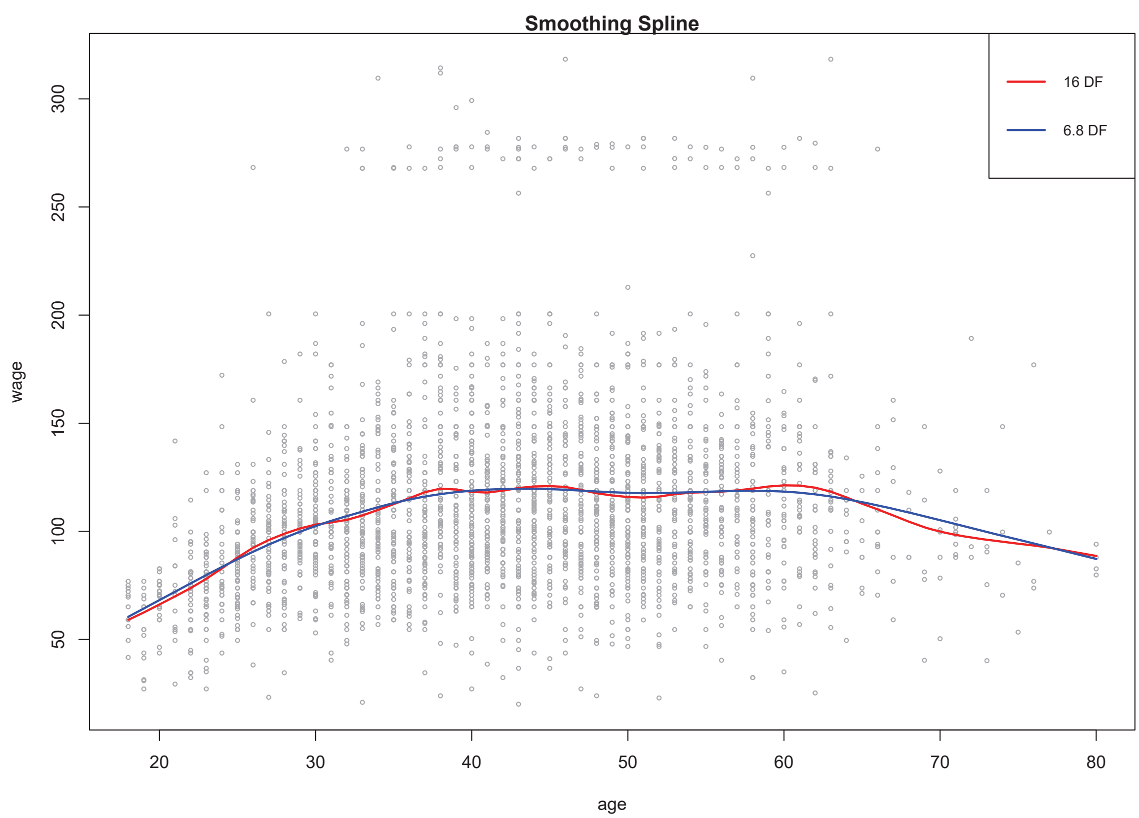
```
> fit2 <- lm(wage ~ ns(age, df = 4), data = Wage)
> pred2 <- predict(fit2, newdata = list(age = age_grid), se = T)
> lines(age_grid, pred2$fit, col = "red", lwd = 2)
```



As with the bs() function, we could instead specify the knots directly using the knots option.

In order to fit a smoothing spline, we use the smooth.spline() function.

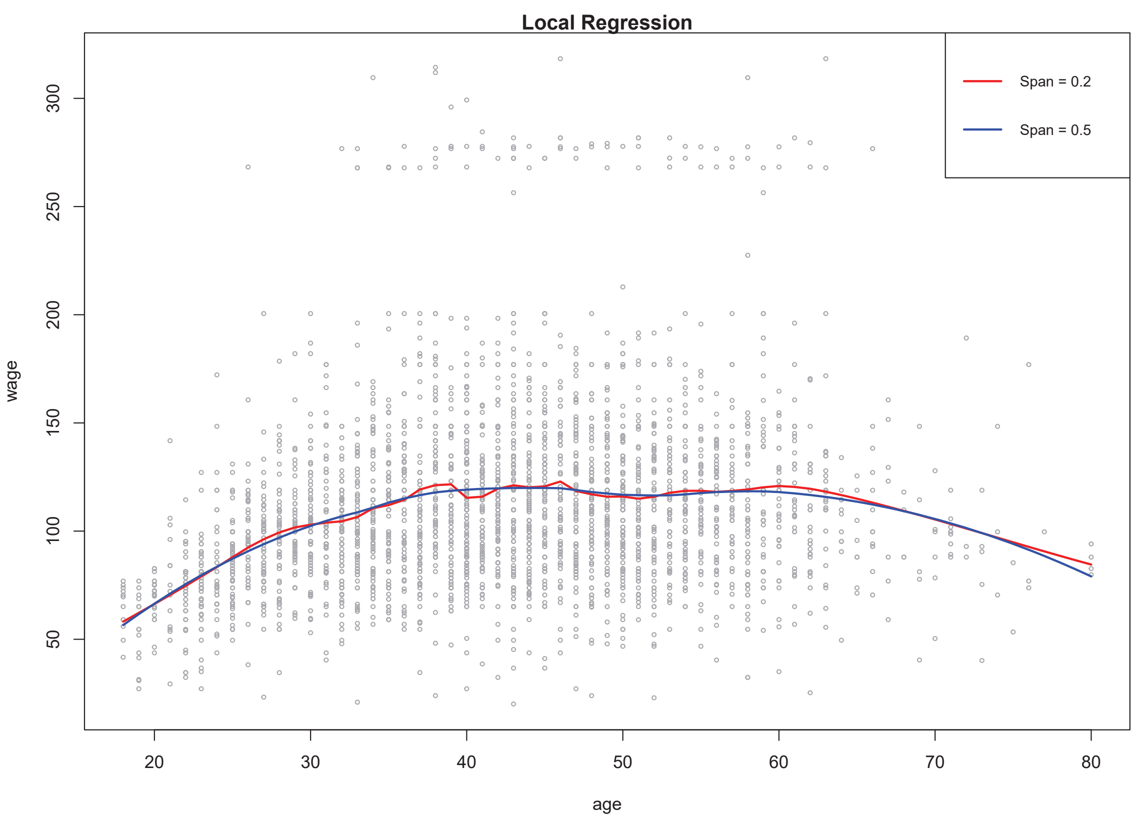
```
> plot(age, wage, xlim = agelims, cex = .5, col = "darkgrey")
> title("Smoothing Spline")
> fit <- smooth.spline(age, wage, df = 16)
> fit2 <- smooth.spline(age, wage, cv = TRUE)
> fit2$df
[1] 6.794596
> lines(fit, col = "red", lwd = 2)
> lines(fit2, col = "blue", lwd = 2)
> legend("topright", legend = c("16 DF", "6.8 DF"),
   col = c("red", "blue"), lty = 1, lwd = 2, cex = .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 cross-validation; this results in a value of λ that yields 6.8 degrees of freedom.

In order to perform local regression, we use the loess() function.

```
> plot(age, wage, xlim = agelims, cex = .5, col = "darkgrey")
> title("Local Regression")
> fit <- loess(wage ~ age, span = .2, data = Wage)
> fit2 <- loess(wage ~ age, span = .5, data = Wage)
> lines(age_grid, predict(fit, 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 = 0.5"),
   col = c("red", "blue"), lty = 1, lwd = 2, cex = .8)
```



Here we have performed local linear regression using spans of 0.2 and 0.5: that is, each neighborhood consists of 20% or 50% of the observations. The larger the span, the smoother the fit.

Generalized Additive Models

- Generalized additive models (GAMs) allow for flexible nonlinearities in several variables, but retains the additive structure of linear models.
- A natural way to extend the multiple linear regression model

  $$y_i = \beta_0 + \beta_1 x_{i1} + \beta_2 x_{i2} + \ldots + \beta_p x_{ip} + \epsilon_i$$

  in order to allow for non-linear relationships between each feature and the response is to replace each linear component $\beta_j x_{ij}$ with a (smooth) non-linear function $f_j(x_{ij})$.
- We would then write the model as

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

- We have discussed several methods for fitting functions to a single variable so far. The beauty of GAMs is that we can use these methods as building blocks for fitting an additive model.
- Take, for example, natural splines, and consider the task of fitting the model

  $$\text{wage} = \beta_0 + f_1(\text{year}) + f_2(\text{age}) + f_3(\text{education}) + \epsilon$$

  on the Wage data. Here year and age are quantitative variables, and education is a qualitative variable with five levels: <HS, HS, <Coll, Coll, >Coll, referring to the amount of high school or college education that an individual has completed.