Polynomial Regression and Basic Splines

Yuan Bian

University of Western Ontario

2022/11/10

Moving Beyond Linearity: $y = f(x) + \epsilon$

- So far, we have mostly focused on linear models.
- Linear models are relatively simple to describe and implement.
- Standard linear regression can have significant limitations in terms of predictive power, due to the linearity assumption.

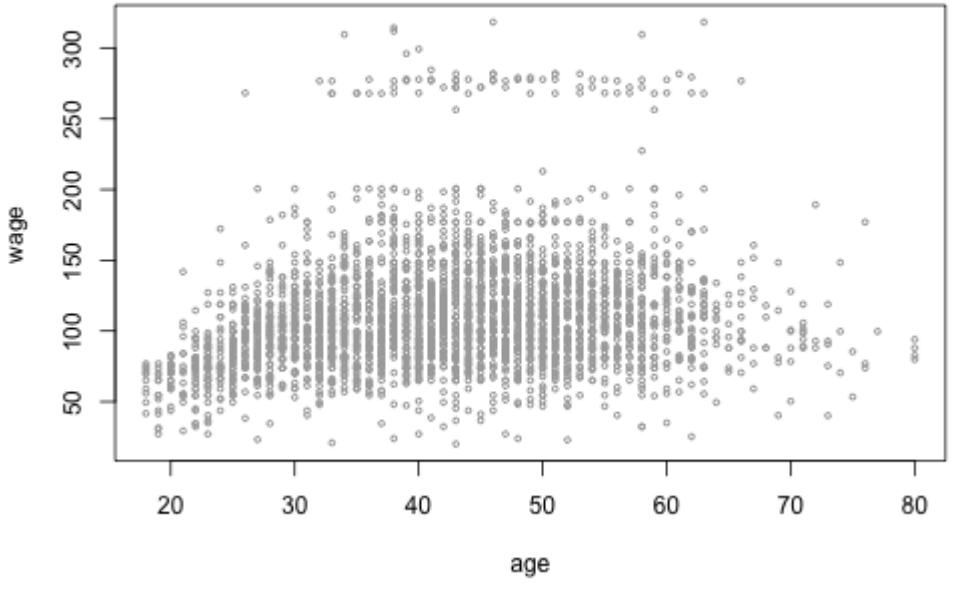
Objective of Today's Session

- We relax the linearity assumption while still attempting to maintain as much interpretability as possible.
- We do this by examining some simple extensions of linear models:
  - **polynomial regression**: extends the linear model by raising the original predictors to a power
  - **regression splines**: more flexible extension of the polynomials

Polynomial Regression

$$f(x) = \beta_0 + \beta_1 x + \beta_2 x^2 + \cdots$$

```
library(ISLR2)
attach(Wage)
agelims <- range(age)
plot(age, wage, xlim = agelims, cex = .5, col = "darkgrey")
```



- The `poly()` command returns a matrix whose columns are a basis of *orthogonal polynomials*

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

```
                Estimate Std. Error t value Pr(>|t|)
(Intercept)        111.7      0.729  153.28 0.00e+00
poly(age, 4)1      447.1     39.915   11.20 1.48e-28
poly(age, 4)2     -478.3     39.915  -11.98 2.36e-32
poly(age, 4)3      125.5     39.915    3.14 1.68e-03
poly(age, 4)4      -77.9     39.915   -1.95 5.10e-02
```

- We can also use `poly()` to obtain `age`, `age^2`, `age^3` and `age^4` directly, by setting `raw = T`.

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

```
                        Estimate Std. Error t value Pr(>|t|)
(Intercept)            -1.84e+02   6.00e+01   -3.07 0.002180
poly(age, 4, raw = T)1  2.12e+01   5.89e+00    3.61 0.000312
poly(age, 4, raw = T)2 -5.64e-01   2.06e-01   -2.74 0.006261
poly(age, 4, raw = T)3  6.81e-03   3.07e-03    2.22 0.026398
poly(age, 4, raw = T)4 -3.20e-05   1.64e-05   -1.95 0.051039
```

There are several other equivalent ways of fitting this model. For example

```
fit2a <- lm(wage ~ age + I(age^2) + I(age^3) + I(age^4), data = Wage)
coef(fit2a)
```

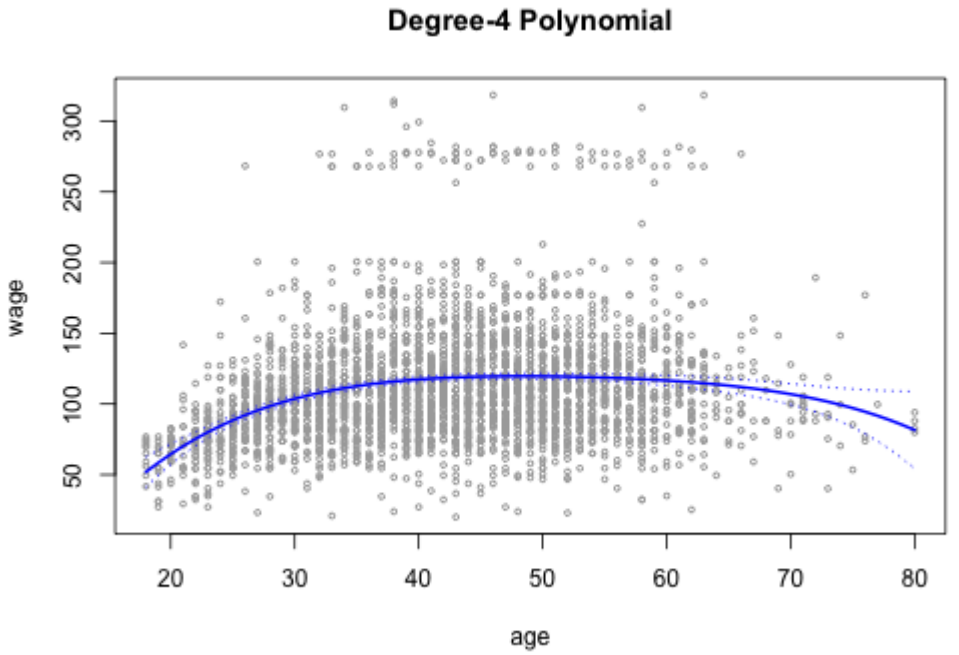
```
(Intercept)          age     I(age^2)     I(age^3)     I(age^4)
  -1.84e+02    2.12e+01   -5.64e-01    6.81e-03   -3.20e-05
```

- the function `I()` protects terms like `age^2` (the `^` symbol has a special meaning in formulas).

```
fit2b <- lm(wage ~ cbind(age, age^2, age^3, age^4), data = Wage)
coef(fit2a)
```

```
(Intercept)          age     I(age^2)     I(age^3)     I(age^4)
  -1.84e+02    2.12e+01   -5.64e-01    6.81e-03   -3.20e-05
```

```r
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)
plot(age, wage, xlim = agelims, cex = .5, col = "darkgrey")
title("Degree-4 Polynomial")
lines(age.grid, preds$fit, lwd = 2, col = "blue")
matlines(age.grid, se.bands, lwd = 1, col = "blue", lty = 3)
```



We mentioned earlier that whether or not an orthogonal set of basis functions is produced in the `poly()` function will not affect the model obtained in a meaningful way.

```
preds2 <- predict(fit2, newdata = list(age = age.grid),
    se = TRUE)
max(abs(preds$fit - preds2$fit))
```

```
[1] 7.82e-11
```

In performing a polynomial regression we must decide on the degree of the polynomial to use. One way to do this is by using hypothesis tests.

```
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)
anova(fit.1, fit.2, fit.3, fit.4, fit.5)
```

```
Analysis of Variance Table

Model 1: wage ~ age
Model 2: wage ~ poly(age, 2)
Model 3: wage ~ poly(age, 3)
Model 4: wage ~ poly(age, 4)
Model 5: wage ~ poly(age, 5)
  Res.Df     RSS Df Sum of Sq      F Pr(>F)
1   2998 5022216
2   2997 4793430  1    228786 143.59 <2e-16 ***
3   2996 4777674  1     15756   9.89 0.0017 **
4   2995 4771604  1      6070   3.81 0.0510 .
5   2994 4770322  1      1283   0.80 0.3697
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

Drawbacks of using **polynomial regression**

- Polynomial basis needs a large number of basis functions in order to show local characteristics of the data.
- Polynomials don't do a good job in the tails.
- They are too sensitive to the outliers.

These drawbacks led to the development of the so called **splines**. Today, we will focus on the polynomial splines with the B-Spline Basis and natural cubic splines. Other popularly used splines are smoothing splines.

- B-splines are basis functions that make the calculation of splines much easier.
- Each B-spline is non-zero in a small interval and zero outside this interval.
- They are formed by polynomial pieces that join each other in the knots; A B-spline of order m can be calculated recursively using the ones with lower order.

More details see De Boor, C. (2001) A Practical Guide to Splines.

Regression Splines

Considering B-splines of order m with k interior knots we can write a function f as:

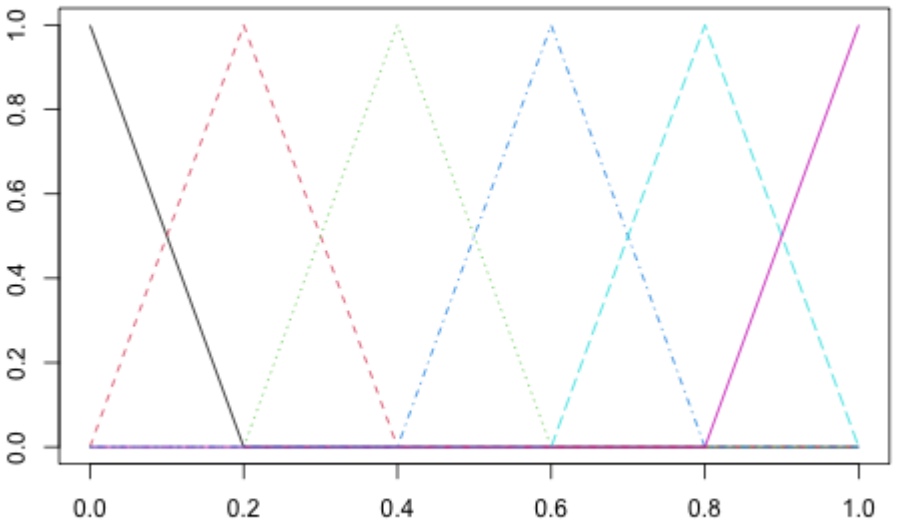
$$f(x) = \sum_{i=1}^{m+k} c_i B_{i,m}(x).$$

In order to use splines, you need to specify any two of the following:

- order m: $m = 2$ (Linear), $m = 3$ (Quadratic), $m = 4$ (Cubic), ...
- # of interior knots k: $k = 1$ (one interior knots at 50th percentiles of x), $k = 2$ (two interior knots at 33.33th and 66.67th percentiles of x) ...; Instead, you can also specify the exact location of the interior knots (place more knots in places where we feel the function might vary rapidly, and place fewer knots where it seems more stable)
- degree of freedom: $df = m + k$

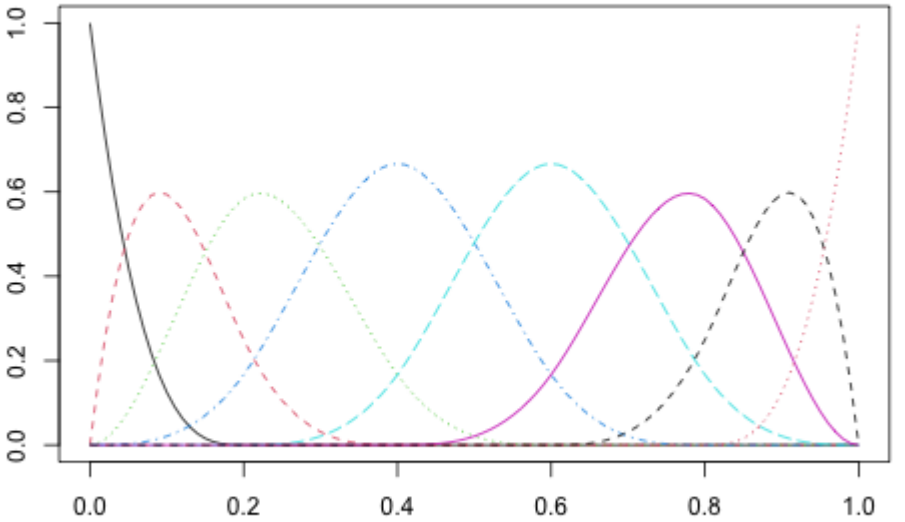
- B-splines of order 2 (piecewise linear)

```
library(splines)
x <- seq(from=0,to=1,by=0.01)
matplot(x, bs(x, df=6, intercept = T, degree=1), type = "l",
        xlab = "", ylab = "")
```

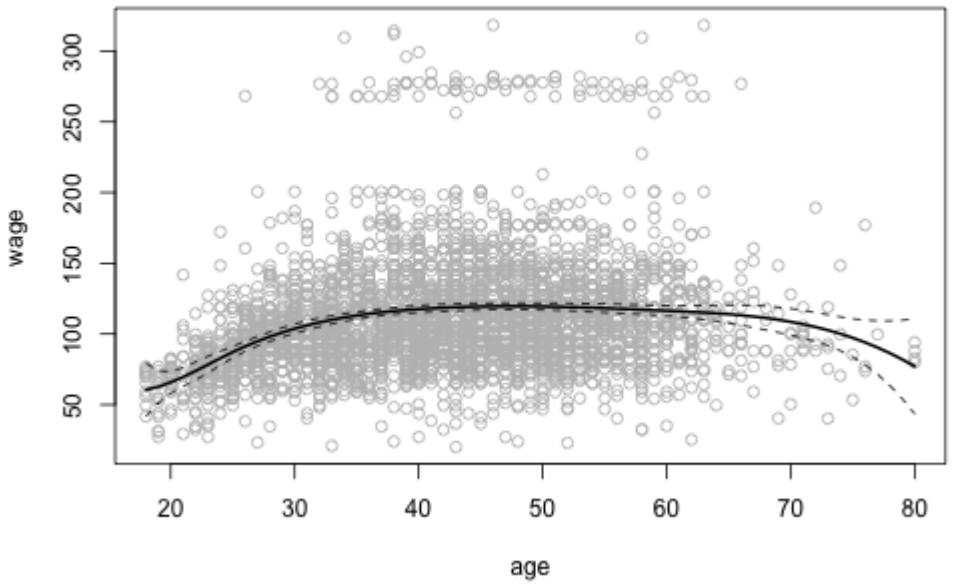


- B-splines of order 4 (piecewise cubic)

```
matplot(x, bs(x, df=8, intercept = T), type = "l",
        xlab = "", ylab = "")
```



```
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.)

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

```
[1] 3000    6
```

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

```
[1] 3000    6
```

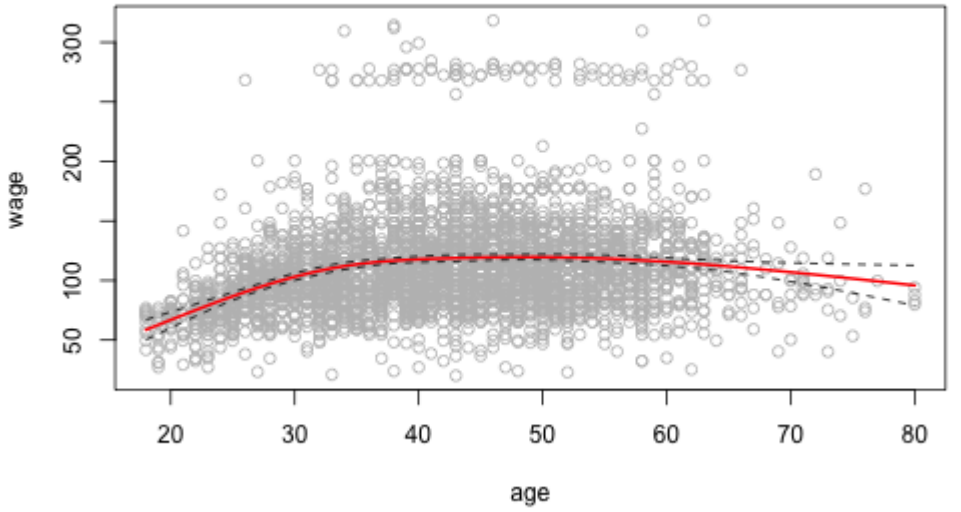
```
attr(bs(age, df = 6), "knots")
```

```
 25%  50%  75%
33.8 42.0 51.0
```

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`.

- Unfortunately, regression splines can have high variance at the boundary of the predictors.
- A **natural spline** is a regression spline with additional boundary constraints to produce more stable estimates at the boundaries.

```
fit2 <- lm(wage ~ ns(age, df = 4), data = Wage)
pred2 <- predict(fit2, newdata = list(age = age.grid), se = T)
plot(age, wage, col = "gray")
lines(age.grid, pred2$fit, col = "red", lwd = 2)
lines(age.grid, pred2$fit + 2 * pred2$se, lty = "dashed")
lines(age.grid, pred2$fit - 2 * pred2$se, lty = "dashed")
```



Reference

James, G, Witten, D, Hastie, T, Tibshirani, R (2013) *An Introduction to Statistical Learning*. Springer, New York, Second edition.