Statistics 452: Statistical Learning and Prediction

Chapter 7, Part 2: Basis Functions and Regression Splines

Brad McNeney

2017-10-24

Basis Functions

- The polynomial of degree 4 that we fit to the wage data is a function $f(X)$ of the form

$$f(X) = \beta_0 + \beta_1 X + \beta_2 X^2 + \beta_3 X^3 + \beta_4 X^4$$

  - Think of this as a linear combination of four functions, $b_1(x) = x, \ldots, b_4(x) = x^4$.
  - The four functions are called *basis functions*

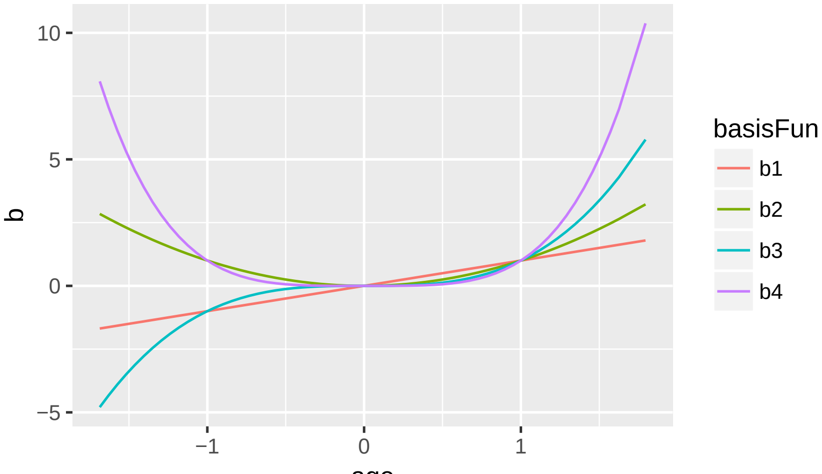
- In this chapter we fit functions $f(X)$ of the form

$$f(X) = \beta_0 + \beta_1 b_1(X) + \ldots, +\beta_k b_K(X)$$

for some choice of basis functions.

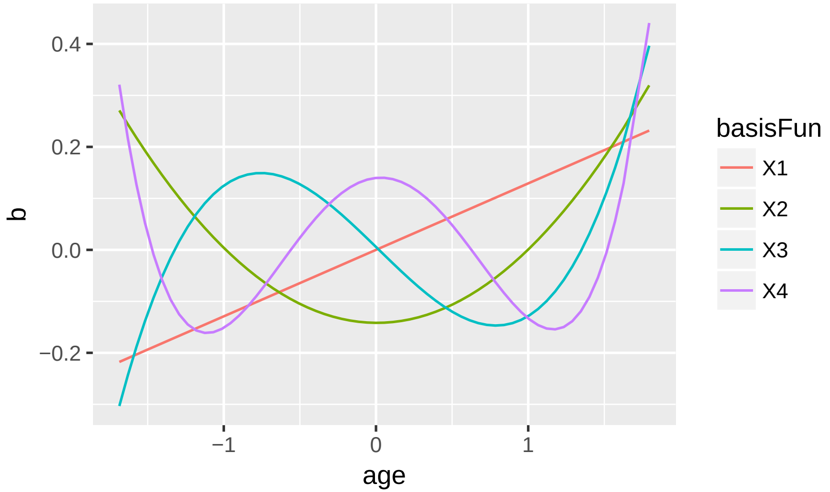
Visualizing Basis Functions

```r
library(ISLR); data(Wage)
age <- sort(unique(Wage$age)); age <- scale(age)
Xmat <- data.frame(age=age,b1=age,b2=age^2,b3=age^3,b4=age^4)
library(tidyr); library(ggplot2)
Xlong <- gather(Xmat,basisFun,b,-age)
ggplot(Xlong,aes(x=age,y=b,color=basisFun)) + geom_line()
```



Basis Functions from poly()

```r
age <- as.numeric(age) #scaling made it unsuitable as input for poly()
Xmat <- data.frame(age=age,poly(age,4))
Xlong <- gather(Xmat,basisFun,b,-age)
ggplot(Xlong,aes(x=age,y=b,color=basisFun)) + geom_line()
```

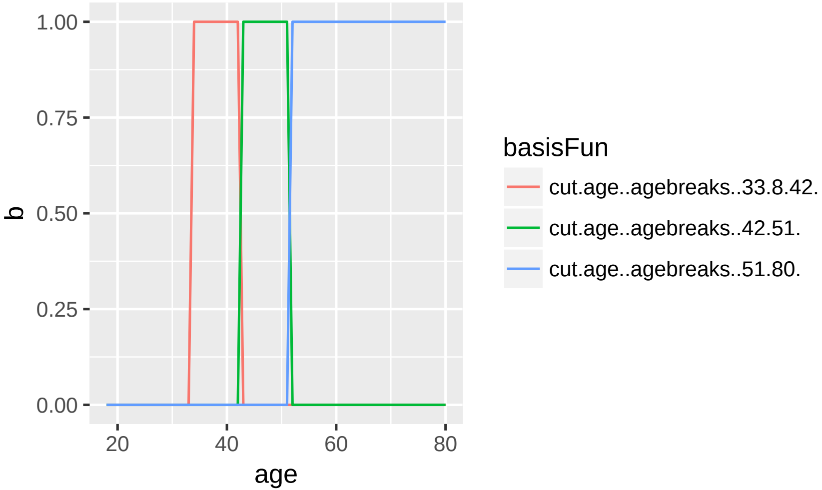


Step Functions as Basis Functions

- The step functions we used to model wages can also be considered basis functions.
- They are defined to be "piece-wise constant"; that is, constant on the separate pieces of the x-axis defined by the age category cut-offs.
  - We will soon refer to the "interior" cut-offs as knots.

```
age <- sort(unique(Wage$age)) # return to original age
# Use min, max and quartiles of age distribution.
agebreaks <- quantile(Wage$age)
# Nuisance: need to widen left-most cutpoint to include min
agebreaks[1]<-agebreaks[1]-1
# Use model.matrix() to set up the dummy vars. Exclude Intercept
Xmat <- model.matrix( ~ cut(age,agebreaks), data=data.frame(age=age))[,-1]
Xmat <- data.frame(age,Xmat)
Xlong <- gather(Xmat,basisFun,b,-age)
```

```r
ggplot(Xlong,aes(x=age,y=b,color=basisFun)) + geom_line()
```

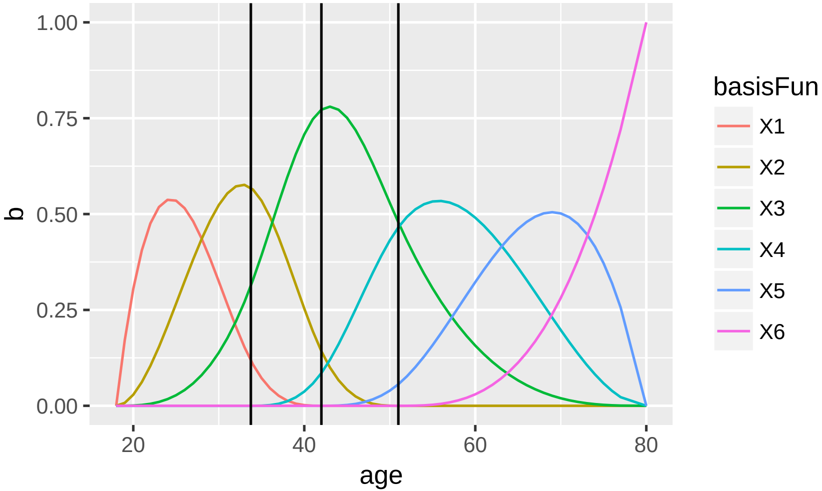


Piecewise Polynomials

- Instead of constants between the cut-points, we can use polynomials.
  - Start with cubic polynomials.
- Then impose constraints on the polynomials that guarantee a smooth function whenever we take a linear combination (more on this later).
  - With these constraints, it turns out that K interior knots leads to $K + 3$ basis functions.
  - For later: $K + 4$ coefficients, or degrees of freedom (df).

```
library(splines)
knots <- agebreaks[2:4] #interior cut-points
Xmat <- data.frame(age=age,bs(age,knots=knots))
Xlong <- gather(Xmat,basisFun,b,-age)
```

```r
ggplot(Xlong,aes(x=age,y=b,color=basisFun)) + geom_line() +
  geom_vline(xintercept=knots)
```



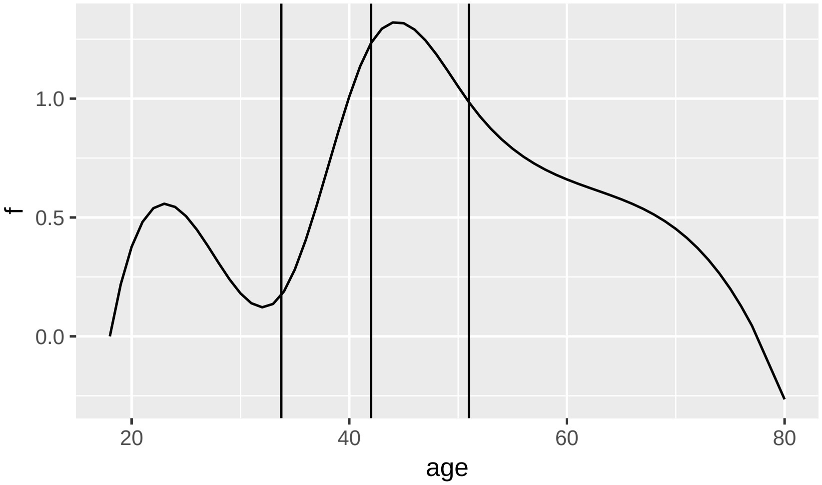
- Note: In the end, the basis functions are not restricted to the intervals between/outside the knots

Constraints Make Smooth Functions

- If we take any linear combination of the basis functions we always get a smooth curve.
  - No jumps, **and** no obvious “kinks”.

```
Xmat <- bs(age,knots=knots)
coefs <- rnorm(ncol(Xmat)) # arbitrary coefficients -- try your own
dat <- data.frame(age=age,f =Xmat%*%coefs)
```

```r
ggplot(dat,aes(x=age,y=f)) + geom_line() +
  geom_vline(xintercept=knots)
```



Fitting a Regression Spline to the Wage Data

- With the basis functions in hand, we use ordinary least squares to fit the regression spline.

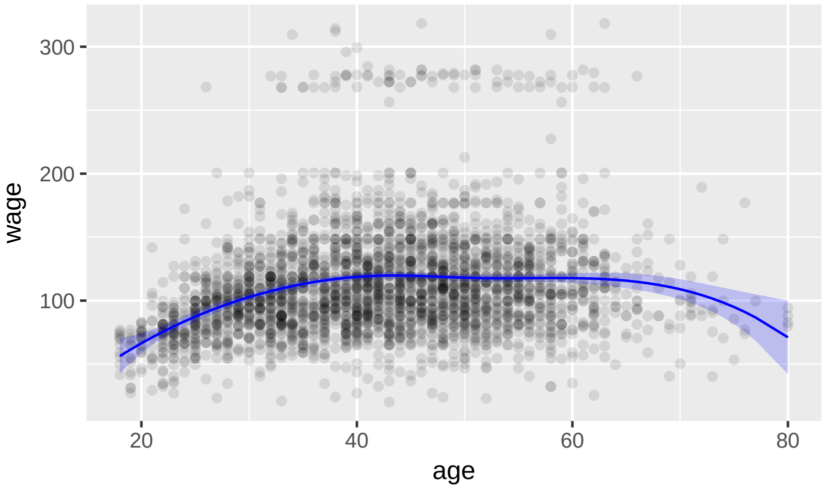
```
fit <- lm(wage ~ bs(age,knots=knots),data=Wage)
summary(fit)$coef
```

```
##                          Estimate Std. Error   t value     Pr(>|t|)
## (Intercept)              56.31384   7.258043 7.7588188 1.167133e-14
## bs(age, knots = knots)1 27.82400  12.434548 2.2376369 2.531805e-02
## bs(age, knots = knots)2 54.06255   7.127490 7.5850746 4.407445e-14
## bs(age, knots = knots)3 65.82839   8.323418 7.9088174 3.623715e-15
## bs(age, knots = knots)4 55.81273   8.723974 6.3976272 1.825054e-10
## bs(age, knots = knots)5 72.13147  13.744891 5.2478751 1.646050e-07
## bs(age, knots = knots)6 14.75088  16.208690 0.9100597 3.628643e-01
```

```
plotfit <- function(fit,dat,newdat){
  preds <- data.frame(newdat,
          predict(fit,newdata=newdat,interval="confidence"))
  ggplot(dat,aes(x=age,y=wage)) + geom_point(alpha=0.1) +
    geom_ribbon(aes(x=age,y=fit,ymin=lwr,ymax=upr),
                data=preds,fill="blue",alpha=.2) +
    geom_line(aes(y=fit),data=preds,color="blue")
}
```

Plotting Fitted Curve

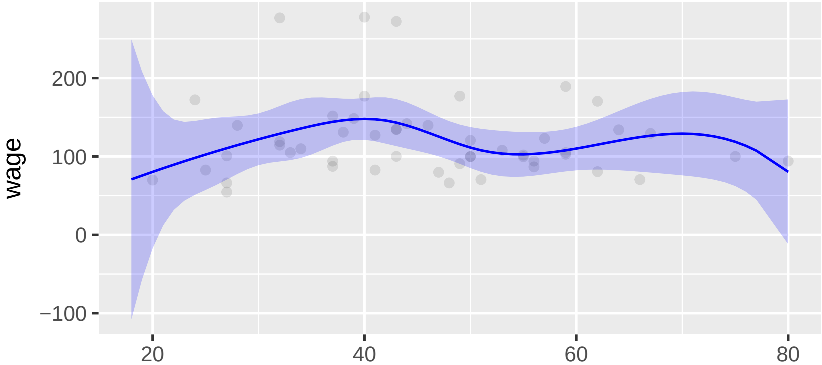
```
newdat <- data.frame(age=age)
plotfit(fit,Wage,newdat)
```



Constraints Outside the Knots

- Estimated splines curves have high variance where data are scarce.
- Illustrate by sampling from the wage data.

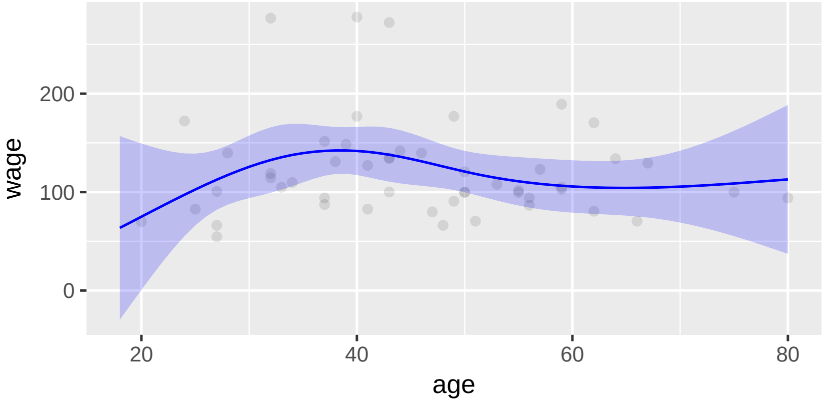
```
set.seed(1)
Wage.sub <- rbind(Wage[sample(1:nrow(Wage),50),c("wage","age")],
                  c(100,75))
fit <- lm(wage ~ bs(age,knots=knots),data=Wage.sub)
plotfit(fit,Wage.sub,newdat)
```



Natural Splines

- Natural splines constrain the function to be linear outside the knots, and these constraint has the effect of reducing variance.
  - Note on df: two extra constraints means two fewer df; i.e., $K+1$.

```
fit <- lm(wage ~ ns(age,knots=knots),data=Wage.sub)
plotfit(fit,Wage.sub,newdat)
```



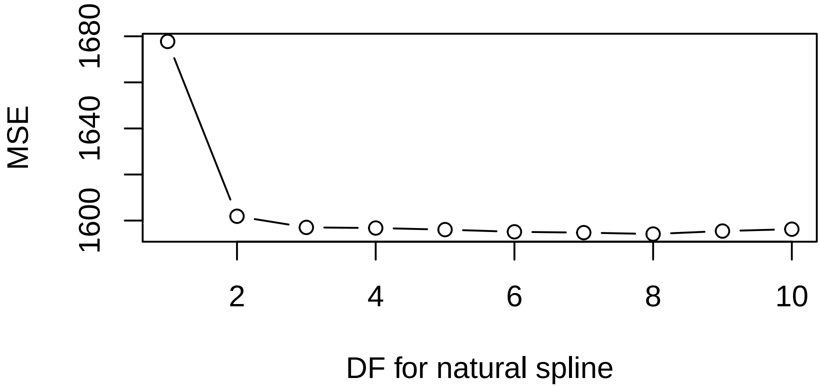
Knots or Degrees of Freedom

- We mentioned that K interior knots leads to $K + 3$ df for the cubic spline and $K + 1$ for the natural spline.
- Can specify df and let `bs()` or `ns()` choose the knots at appropriate quantiles of `age`.
- Why? A single smoothing parameter lends itself to cross-validation to select it.

```
cv <- function(df,k,seed) {
  # Setup
  set.seed(seed)
  folds <- sample(1:k,size=nrow(Wage),replace=TRUE)
  validErr <- rep(NA,k)
  # Loop over folds: train, test, MSE
  for(i in 1:k) {
    testWage <- Wage[folds==i,]
    trainWage <- Wage[folds!=i,]
    fit <- lm(wage ~ ns(age,df=df),data=trainWage)
    testPreds <- predict(fit,testWage)
    validErr[i] <- mean((testPreds - testWage$wage)^2)
  }
  mean(validErr)
}
k<-10; nDf <- 10; seed <- 1; cvErrs <- rep(NA,nDf)
for(df in 1:nDf){ # loop over df
  cvErrs[df] <- cv(df,k,seed)
}
cvErrs
```

```
##  [1] 1677.760 1601.883 1597.044 1596.767 1596.078 1595.124 1594.772
##  [8] 1594.168 1595.454 1596.263
```

```r
plot(1:nDf,cvErrs,type="b",ylab="MSE",xlab="DF for natural spline")
```



Conclusion for Wage Data

- About three df looks adequate.

```
fit <- lm(wage ~ ns(age,df=3),data=Wage)
plotfit(fit,Wage,newdat)
```