Splines

Trevor Hastie and Robert Tibshirani

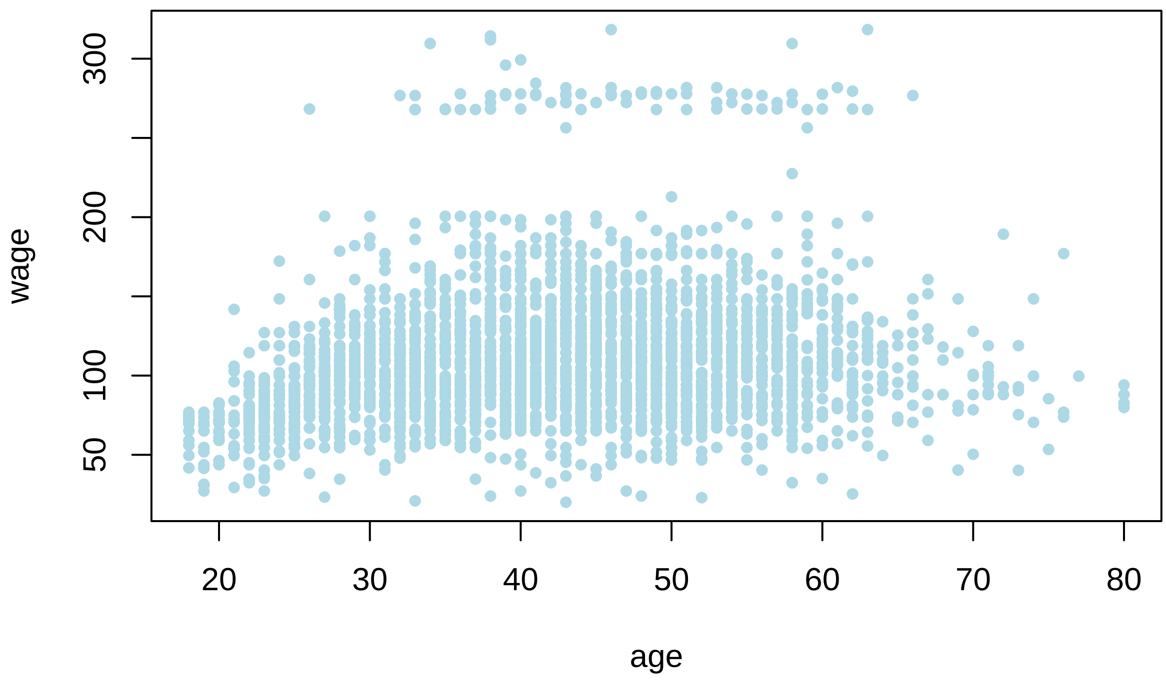
Here, I am adapting part of the lab associated with Chapter 7 of the textbook.

In this lab, we re-analyze the `Wage` data considered in the examples throughout this chapter, in order to illustrate the fact that many of the complex non-linear fitting procedures discussed can be easily implemented in `R`. We begin by loading the `ISLR2` library, which contains the data.

```
library(ISLR2)
attach(Wage)
dim(Wage)
```

```
## [1] 3000   11
```

```
plot(age, wage,
     pch=20, col="lightblue")
```



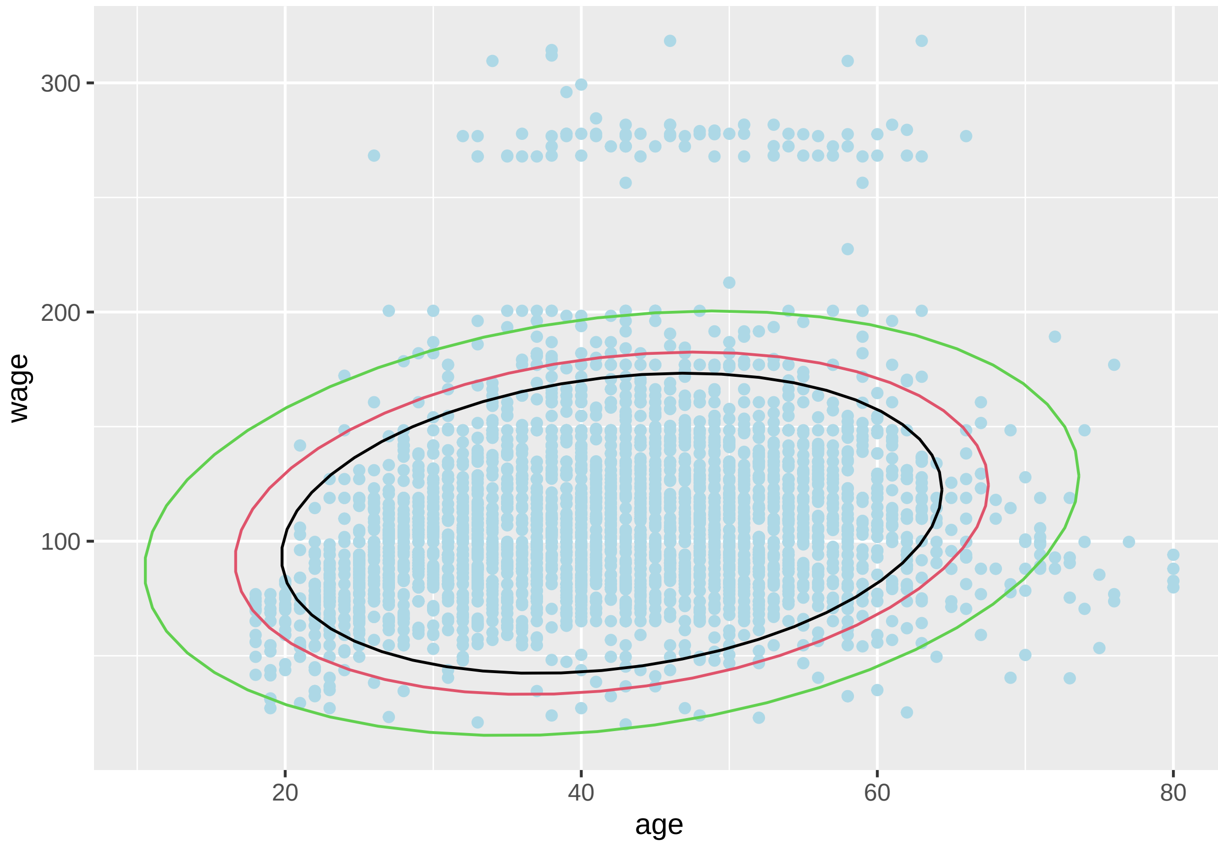
I am inserting some more exploratory data analysis here based on student feedback.

```
library(ggplot2)

df=data.frame(age,wage)

ggplot(df, aes(x = age, y = wage)) +
  geom_point(color="lightblue") +
  stat_ellipse(level = 0.9) +
  stat_ellipse(level = 0.95, color = 2) +
```

```
  stat_ellipse(level = 0.99, color = 3)
```



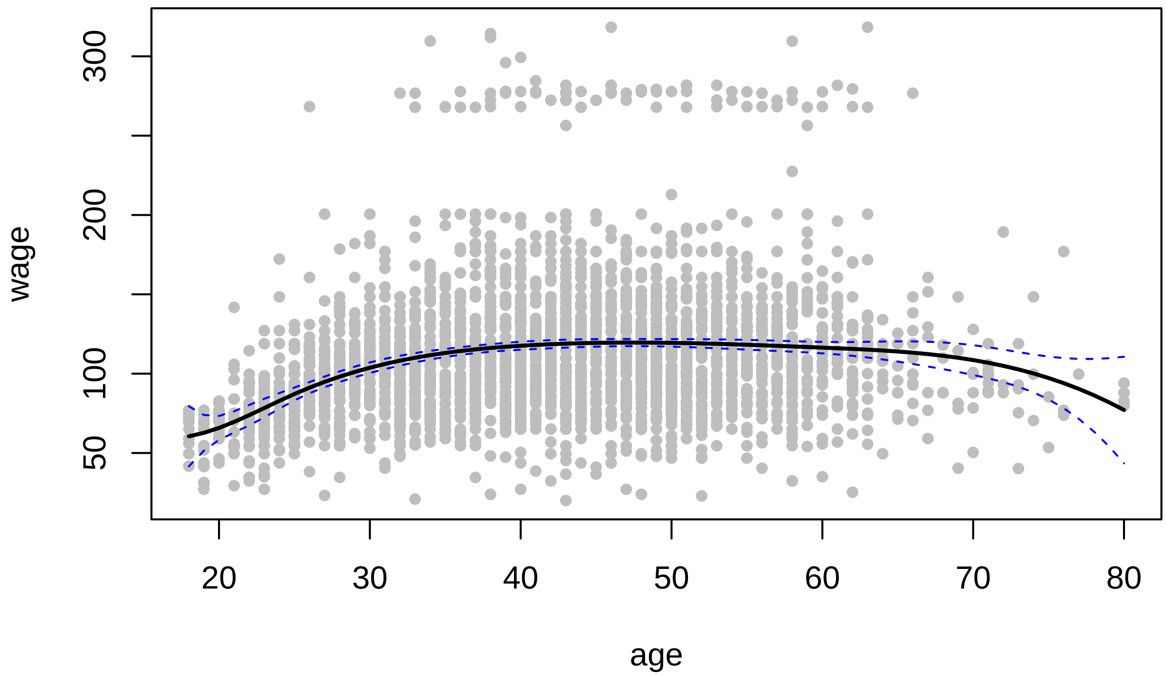
Cubic splines

In order to fit regression splines in R, we use the `splines` library. In Section 7.4, we saw that regression splines can be fit by constructing an appropriate matrix of basis functions. 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)

agelims <- range(age)
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)

pred <- predict(fit, newdata = list(age = age.grid), se = T)
plot(age, wage, col = "gray", pch=20)
lines(age.grid, pred$fit, lwd = 2)
lines(age.grid, pred$fit + 2 * pred$se, lty = "dashed", col="blue")
lines(age.grid, pred$fit - 2 * pred$se, lty = "dashed", col="blue")
```



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.) 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
```

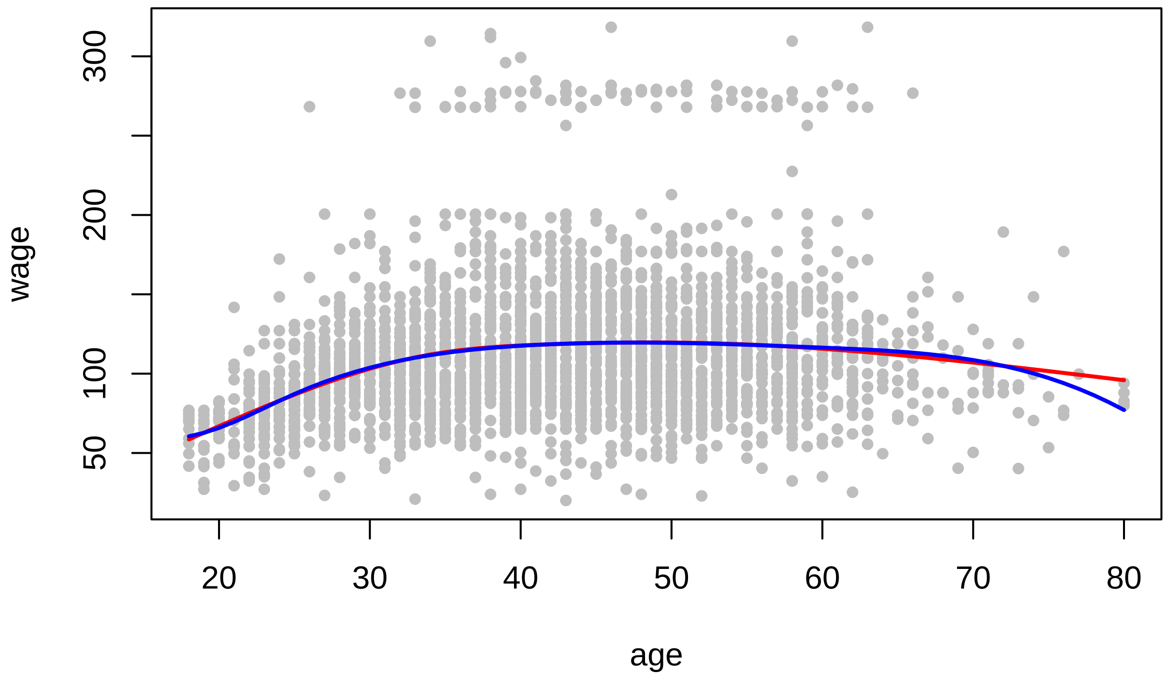
```
attr(bs(age, df = 6), "knots")
```

```
## [1] 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)
plot(age, wage, col = "gray", pch=20)
lines(age.grid, pred2$fit, col = "red", lwd = 2)
lines(age.grid, pred$fit, lwd = 2, col="blue")
```



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. Figure 7.8 was produced with the following code:

```
plot(age, wage, xlim = agelims, cex = .5, col = "darkgrey",
     pch=20)
title("Smoothing Spline")
fit <- smooth.spline(age, wage, df = 16)
fit2 <- smooth.spline(age, wage, cv = TRUE)
```

```
## Warning in smooth.spline(age, wage, cv = TRUE): cross-validation with
## non-unique 'x' values seems doubtful
```

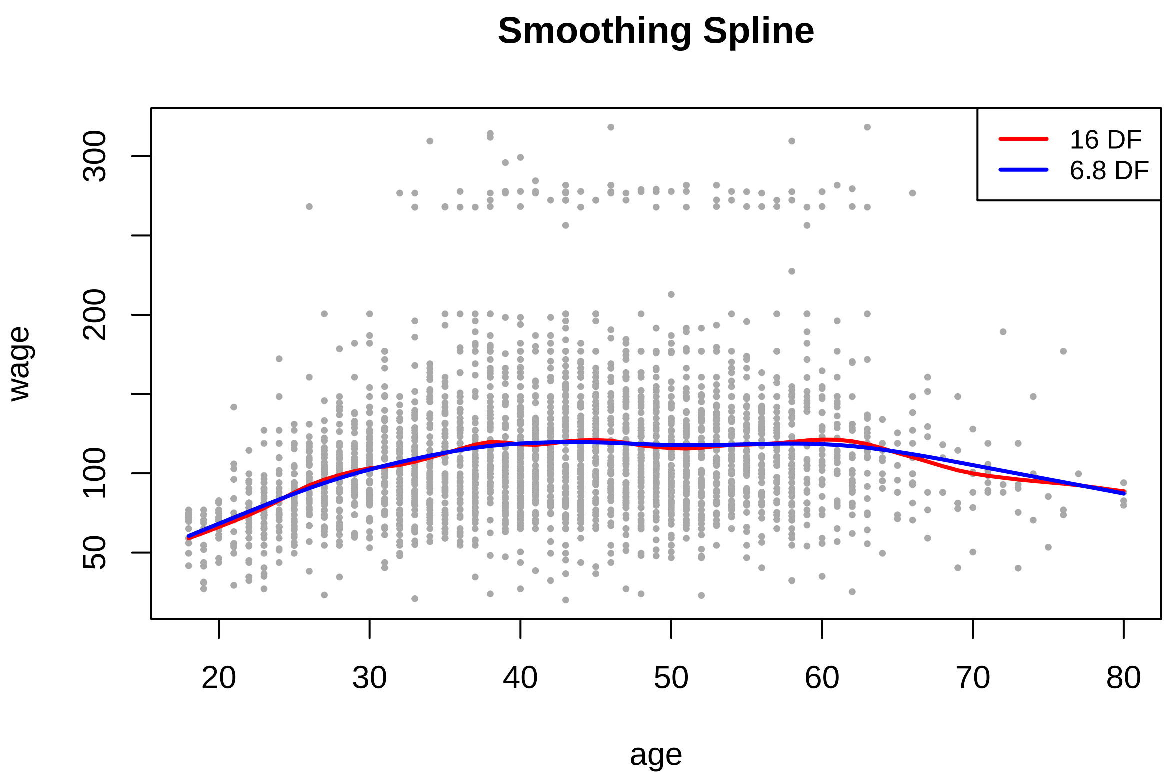
```
fit2
```

```
## Call:
## smooth.spline(x = age, y = wage, cv = TRUE)
##
## Smoothing Parameter  spar= 0.6988943  lambda= 0.02792303 (12 iterations)
## Equivalent Degrees of Freedom (Df): 6.794596
## Penalized Criterion (RSS): 75215.9
## PRESS(l.o.o. CV): 1593.383
```

```
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.

** How do we create confidence intervals here? In this case, the preferred technique is to generate Bayesian "confidence intervals" using bootstrap**

** Now, here is the last spline in `ggplot2` with "confidence intervals". **

```
library(ggformula)
```

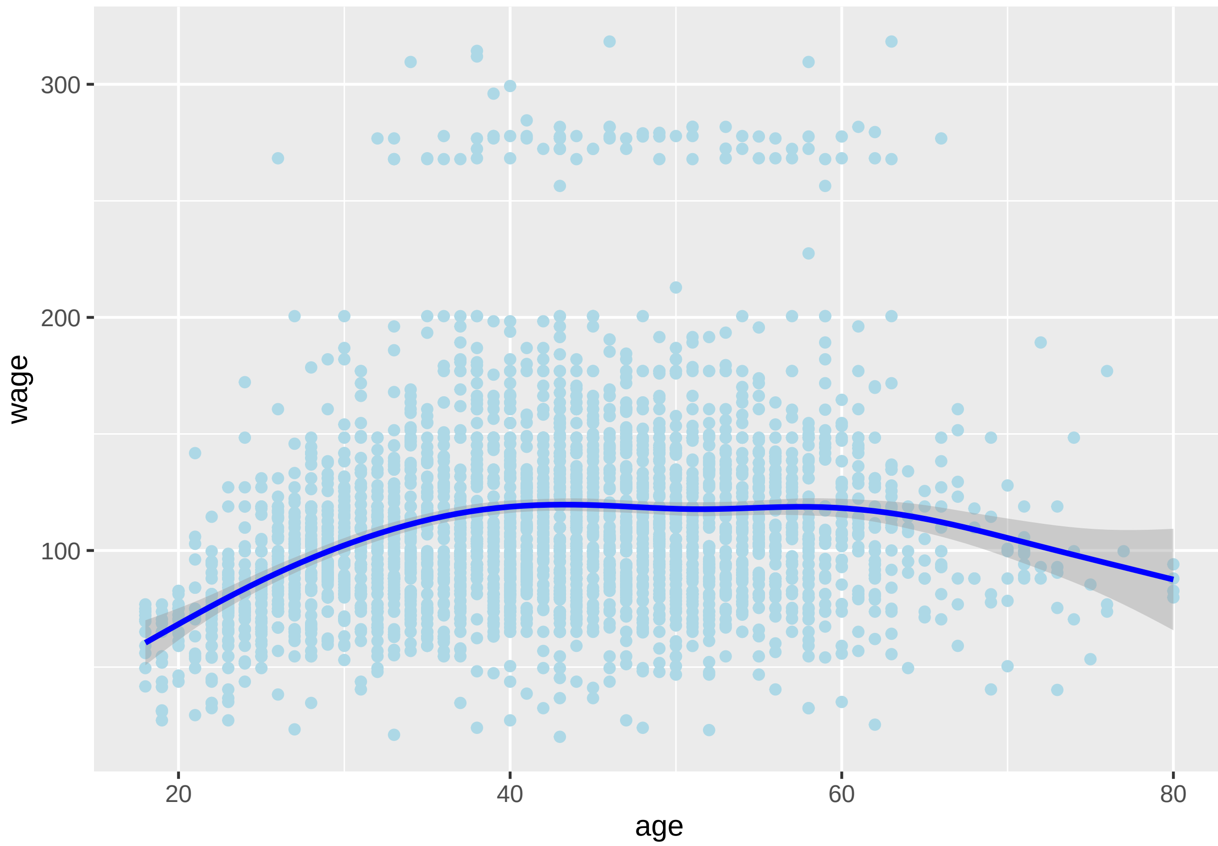
```
## Loading required package: scales

## Loading required package: ggridges

##
## New to ggformula?  Try the tutorials:
##  learnr::run_tutorial("introduction", package = "ggformula")
##  learnr::run_tutorial("refining", package = "ggformula")
```

```
ggplot(df, aes(x = age, y = wage)) +
  geom_point(color="lightblue") +
  stat_smooth(color="blue")
```

```
## `geom_smooth()` using method = 'gam' and formula = 'y ~ s(x, bs = "cs")'
```


300
200
100
wage
20
40
60
80
age