MATH5824 weekly exercises (with solutions)

Dr Stuart Barber

May 2022

These questions are intended to be worked on alongside reading the notes. To help keep track of where we are up to, they are separated by week they were set.

Week 2

1. *In section 2.1 of the notes, what are $f^{(p)}(t_i-)$ and $f^{(p)}(t_i+)$ in terms of the coefficients $\{a_{ij}\}$? What would happen if we specified additional constraints $f^{(p)}(t_i-) = f^{(p)}(t_i+)$ for $i = 1, \ldots, n$?*

   Recall that $f(t) = \sum_{j=0}^{p} a_{ij} t^j$, so $f^{(p)}(t) = p! a_{ip}(t)$ for $t \in [t_i, t_{i+1})$. Requiring continuity at the knots imposes n extra constraints, so reducing the degrees of freedom from $n + p + 1$ to $p + 1$ — the same as a normal polynomial of degree p, and unaffected by the number of knots.

Week 4

2. *Give a self-contained definition of a natural spline.*

   A natural spline of order p (where p must be a positive odd integer) with knots $t_1 < \cdots < t_n$ is a piecewise function which is a polynomial of order p in each interval $[t_i, t_{i+1}); i = 1, \ldots, n-1$ and satisfies the conditions

   $$f^{(l)}(t_i-) = f^{(l)}(t_{i+1}+) \quad \text{for} \quad l = 0, \ldots, p-1$$

   and

   $$f^{(l)}(t) = 0 \quad \text{for} \quad l = (p+1)/2, \ldots, p \text{ and } t < t_1 \text{ or } t > t_n.$$

3. *Let*

   $$f(t) = 3 + 2t + 4|t|^3 + |t-1|^3.$$

   Write f as a cubic polynomial in each of the intervals $(-\infty, 0)$, $(0, 1)$ and $(1, \infty)$. Verify that f and its first two derivatives are continuous at the knots.

   Is f a spline? Is f a natural spline?

Expand $f(t) = 3 + 2t + 4|t|^3 + |t-1|^3$ as a cubic on each of the three intervals and compute derivatives to get

$$\begin{aligned} f(t) &= 3 + 2t - 4t^3 - (t-1)^3 = -5t^3 + 3t^2 - t + 4 \qquad (-\infty, 0) \\ f'(t) &= -15t^2 + 6t - 1 \\ f''(t) &= -30t + 6 \\ f'''(t) &= -30 \end{aligned}$$

$$\begin{aligned} f(t) &= 3 + 2t + 4t^3 - (t-1)^3 = 3t^3 + 3t^2 - t + 4 \qquad (0, 1) \\ f'(t) &= 9t^2 + 6t - 1 \\ f''(t) &= 18t + 6 \\ f'''(t) &= 18 \end{aligned}$$

$$\begin{aligned} f(t) &= 3 + 2t + 4t^3 + (t-1)^3 = 5t^3 - 3t^2 + 5t + 2 \qquad (1, \infty) \\ f'(t) &= 15t^2 - 6t + 5 \\ f''(t) &= 30t - 6 \\ f'''(t) &= 30. \end{aligned}$$

Evaluating left and right limits at the knots of f and its derivatives yields

$$\begin{aligned} &f(0-) = f(0+) = 4, \quad f'(0-) = f'(0+) = -1, \quad f''(0-) = f''(0+) = 6, \\ &\text{but } -30 = f'''(0-) \neq f'''(0+) = 18, \end{aligned}$$

$$\begin{aligned} &f(1-) = f(1+) = 9, \quad f'(1-) = f'(1+) = 14, \quad f''(1-) = f''(1+) = 24, \\ &\text{but } 18 = f'''(1-) \neq f'''(1+) = 30. \end{aligned}$$

That is, f and its first two derivatives are continous at the knots, but not the third derivatives. Further f is a piecewise cubic function since it is cubic in each of the three intervals. Hence f is a cubic spline.

However, f is not a natural spline since it is not a linear function of t for $t < 0$ and $t > 1$.

Another way to prove this result is to write f in the form $f(t) = a_0 + a_1 t + \sum_{j=1}^{n} b_j |t - t_j|^3$, where $n = 2$ and $b_1 = 4$, $b_2 = 1$. Since the function $|t - t_j|^3$ is twice continuously differentiable at $t = t_j$ (check) and is cubic for $t < t_j$ and for $t > t_j$, it follows immediately that f is a cubic spline. But since it is not the case that $\sum b_j = 0$, $\sum b_j t_j = 0$, it follows that the spline is not natural.

4. *Let*

$$f(t) = 3 + |t| - |t - 2|.$$

Show by direct calculation that

$$\int_{-\infty}^{\infty} \{f'(t)\}^2 \, dt = 8.$$

Show that this integral can also be written in the form $-2\mathbf{b}^T K\mathbf{b}$*, where you should define* $\mathbf{b}$ *and* K*.*

The function $f(t) = 3 + |t| - |t-2|$ takes the more explicit form

$$\begin{aligned} f(t) &= 3 - t + (t-2) = 1, \quad t < 0, \\ f(t) &= 3 + t + (t-2) = 2t + 1, \quad 0 < t < 1, \\ f(t) &= 3 + t - (t-2) = 5, \quad t > 2. \end{aligned}$$

Note that f is a natural linear spline (natural since f is constant for $t < 0$ and $t > 2$). Its first derivative takes the form $f'(t) = 0,\ t < 0, f'(t) = 2,\ 0 < t < 2$, and $f'(t) = 0,\ t > 2$. Hence

$$\int_{-\infty}^{\infty} \{f'(t)\}^2\, dt = \int_0^2 2^2\, dt = 8.$$

In terms of the 2×2 matrix $K = (k_{ij}) = (-2|t_i - t_j|)$, where $t_1 = 0,\ t_2 = 2$ and the coefficient vector $\mathbf{b}$ with $b_1 = 1,\ b_2 = -1$, we find that the quadratic form

$$\mathbf{b}^T K\mathbf{b} = \begin{bmatrix} 1 & -1 \end{bmatrix} \begin{bmatrix} 0 & -4 \\ -4 & 0 \end{bmatrix} \begin{bmatrix} 1 \\ -1 \end{bmatrix} = 8$$

takes the same value as the integral.

Week 5

5. *Check that equation (28) in the notes can recover the coefficients of the linear spline in the R script from lecture 2. That is, if a natural linear spline* $f(t)$ *takes values* $4, -1, 1$ *at knots* $1, 2, 3$*, show that* $f(t) = 2.5 - 2.5|t-1| + 3.5|t-2| - |t-3|$*.*

   Equation (28) says

$$\begin{bmatrix} \mathbf{b} \\ a_0 \end{bmatrix} = \{M^{(1)}\}^{-1} \begin{bmatrix} \mathbf{y} \\ 0 \end{bmatrix}.$$

   In this case, $\mathbf{y} = (4, -1, 2)^T$ and

$$M^{(1)} = \begin{bmatrix} K^{(1)} & L^{(1)} \\ L^{(1)T} & 0 \end{bmatrix} = \begin{bmatrix} 0 & 1 & 2 & 1 \\ 1 & 0 & 1 & 1 \\ 2 & 1 & 0 & 1 \\ 1 & 1 & 1 & 0 \end{bmatrix}.$$

   We can evaluate this in R with the following code.

```
K = matrix(c(0, 1, 2, 1, 0, 1, 2, 1, 0), byrow=T, ncol=3)
M = rbind(cbind(K, 1), c(1,1,1,0))
M
```

```
##      [,1] [,2] [,3] [,4]
## [1,]    0    1    2    1
```

```
## [2,]    1    0    1    1
## [3,]    2    1    0    1
## [4,]    1    1    1    0
```

```
y = c(4, -1, 1)
solve(M) %*% c(y, 0)
```

```
##      [,1]
## [1,] -2.5
## [2,]  3.5
## [3,] -1.0
## [4,]  2.5
```

We see that this recovers the original coefficients $a = 2.5, b_1 = -2.5, b_2 = 3.5, b_3 = -1$ as originally specified (note that a_0 is the last element in the vector, although it usually comes first when we write out the spline as a function of t).

6. *(Harder: Use R try to find natural linear and cubic splines which interpolate the beetle data.)*

   We can find these splines as follows. The key points are construction of the matrices K, L and hence M. In the previous question, we wrote them out "by hand", but in general this is not practical. First load the data:

```
beetle = read.table("beetle.txt", header=T)
dose = beetle$dose
mort = beetle$died/beetle$total
n = length(dose)
```

Now define a function to evaluate the absolute difference between two points and apply this to the knots to construct $K^{(1)}$ and hence $M^{(1)}$, which can then be used tofind the coefficients of the linear spline.

```
absdiff = function(x,y){abs(x-y)}
K1 = outer(dose, dose, absdiff)
M1 = rbind(cbind(K1, rep(1,n)), c(rep(1,n),0))
lin.coef = solve(M1) %*% c(mort, 0)
```

We can repeat the process with a function that computes the cube of the absolute differe

```
absdiff3 = function(x,y){abs(x-y)^3}
K2 = outer(dose, dose, absdiff3)
L2 = cbind(rep(1, n), dose)
M2 = rbind(cbind(K2, L2), cbind(t(L2),matrix(0, nrow=2, ncol=2)))

coef3 = solve(M2) %*% c(mort, 0, 0)
```

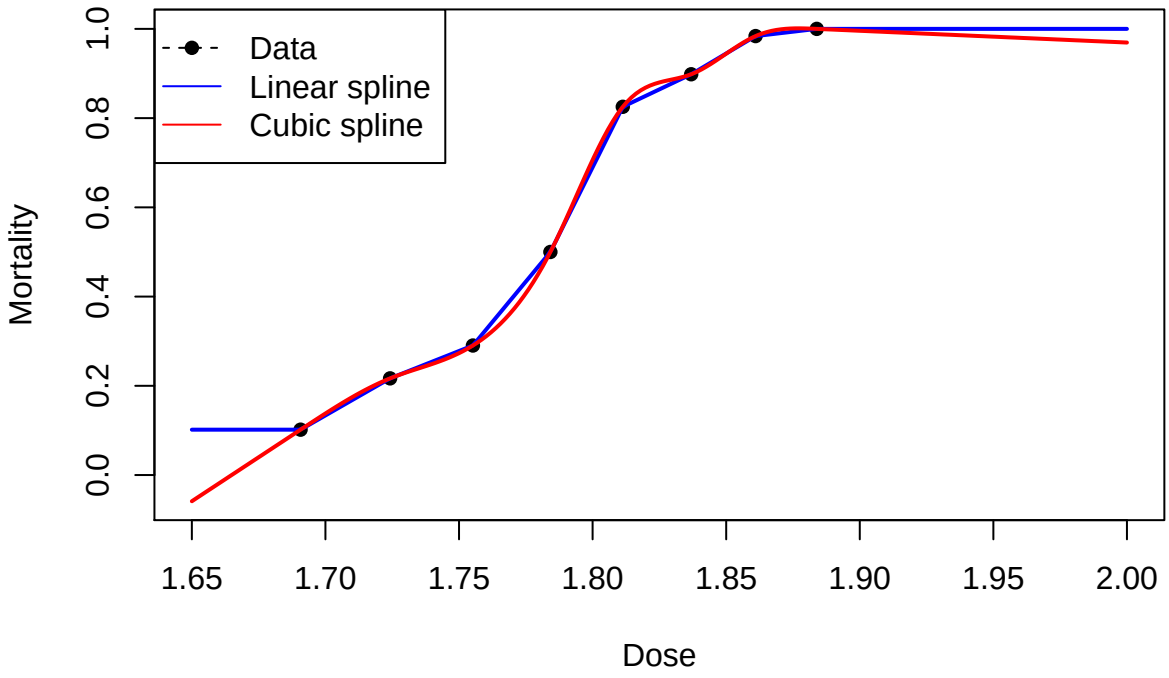
We now have the coefficients to define both splines. To display them, the easiest way is to create a grid of values of `dose`, evaluate the splines at these points, and then plot the results.

```
nat.lin.spline = function(x, coef, knots){
    ## Evaluate linear spline with specified coefficients and knots at
    ## the point x
    return(coef[length(coef)] + sum(coef[-length(coef)] * abs(x - knots)))
}

nat.cubic.spline = function(x, coef, knots){
    ## Evaluate cubic spline with specified coefficients and knots at
    ## the point x
    a0 = coef[length(coef)-1]
    a1 = coef[length(coef)]
    b = coef[1:(length(coef)-2)]
    return(a0 + a1*x + sum(b * abs(x - knots)^3))
}

## Compute both splines on a grid of x values, then plot the results.
xx = seq(from=1.65, to=2.0, length=201)
yy.lin = yy.cub = numeric(length(xx))
for(i in 1:201){
    yy.lin[i] = nat.lin.spline(xx[i], lin.coef, dose)
    yy.cub[i] = nat.cubic.spline(xx[i], coef3, dose)
    }

plot(mort~dose, pch=16, type="b", lty=2, xlim=range(xx),
     ylim=range(yy.lin, yy.cub), xlab="Dose", ylab="Mortality")
lines(yy.lin~xx, col="blue", lwd=2)
lines(yy.cub~xx, col="red", lwd=2)
legend(x = "topleft", pch=c(16, NA, NA), lty=c(2,1,1),
       col=c("black", "blue", "red"),
       legend = c("Data", "Linear spline", "Cubic spline"))
```



As expected, both splines interpolate the data points. Notice that the dashed "data" line is invisible since it is hidden below the linear spline. Notice also that the splines display the characteristic natural spline behaviour; a horizontal line for the linear spline (first derivative is zero), and a straight line for the cubic spline (second derivative is zero). These are the constraints we applied to limit the interpolating splines to be natural interpolating splines.

Week 8

7. *Consider the 'Old Faithful' data set on geyser eruptions available in R. Use the following commands to learn more about and visualise the data:*

   ```
   data(faithful)
   help(faithful)
   plot(faithful)
   ```

   Fit a cubic smoothing spline to these data, using a range of values for the smoothing parameter λ. By eye, suggest a suitable value for λ.

   First, define functions to fit and evaluate the cubic splines.

```
## Function to find the coefficients a and b for a given data set (x,y)
## and lambda
get.coef = function(x, y, lambda){
```

```
    n = length(x)
    absdiff3 = function(x,y){abs(x-y)^3}
    K = outer(x, x, absdiff3) + lambda * diag(n)
    L = cbind(rep(1, n), x)
    M = rbind(cbind(K, L), cbind(t(L),matrix(0, nrow=2, ncol=2)))

    coef = solve(M) %*% c(y, 0, 0)

    return(coef)
}

## Function to evaluate natural cubic spline at location x given
## coefficients and knots.
nat.cubic.spline = function(x, coef, knots){
    a0 = coef[length(coef)-1]
    a1 = coef[length(coef)]
    b = coef[1:(length(coef)-2)]
    return(a0 + a1*x + sum(b * abs(x - knots)^3))
}
```

Now, load and inspect the data.

```
data(faithful)
help(faithful)
attach(faithful)
```

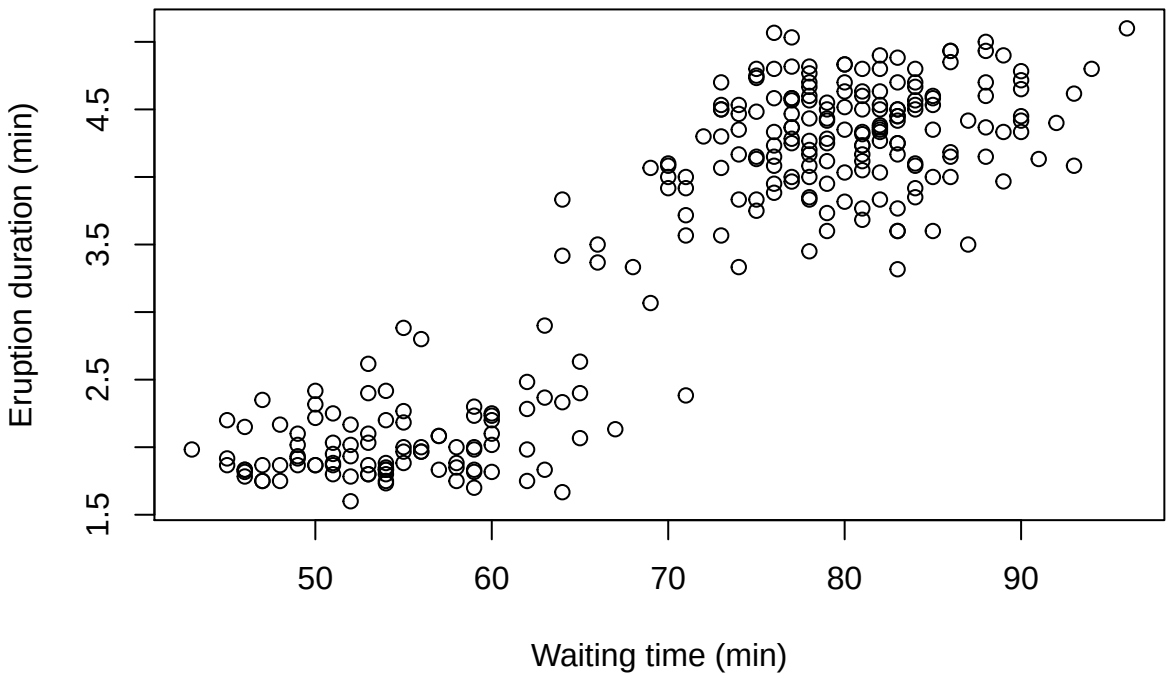
```
## The following objects are masked from faithful (pos = 3):
##
##     eruptions, waiting

## The following objects are masked from faithful (pos = 4):
##
##     eruptions, waiting

## The following objects are masked from faithful (pos = 5):
##
##     eruptions, waiting

## The following objects are masked from faithful (pos = 6):
##
##     eruptions, waiting
```

```
plot(x=waiting, y=eruptions, xlab = "Waiting time (min)",
     ylab = "Eruption duration (min)")
```



We can see that the eruption times seem to be divided into two groups depending on waiting times, but within each group there is little if any association.

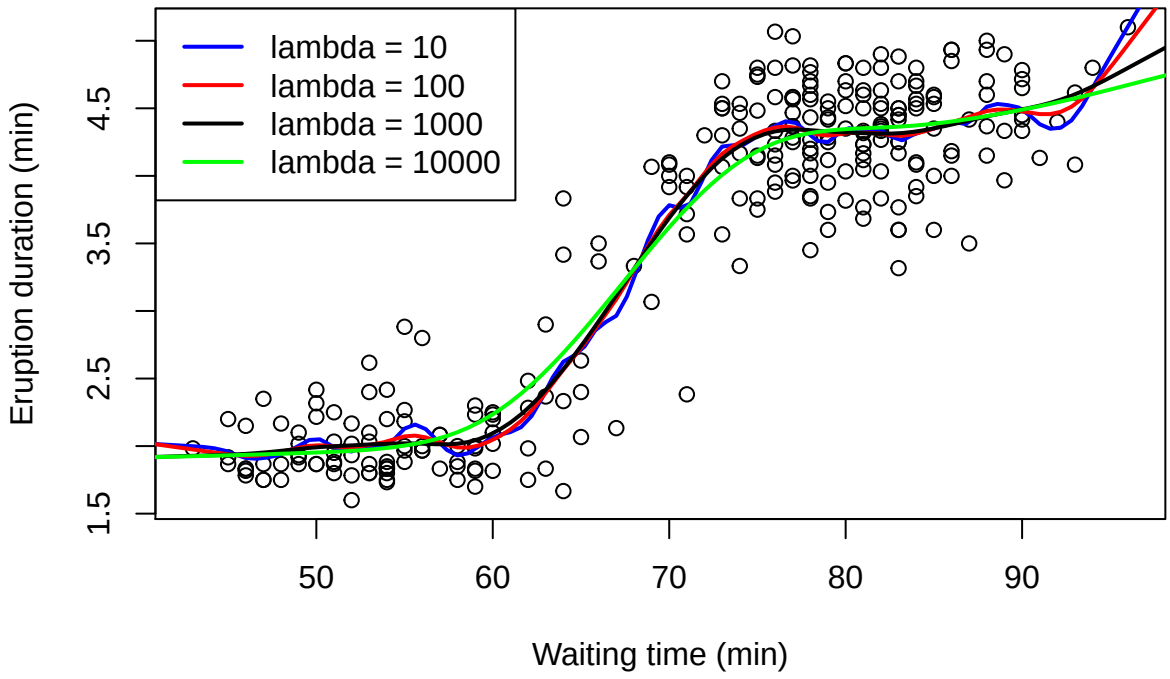
Now we fit cubic smoothing splines with $\lambda = 10, 100, 1000, 10000$ and inspect the results.

```
## Fit and plot spline

plot(x=waiting, y=eruptions, xlab = "Waiting time (min)",
     ylab = "Eruption duration (min)")
f.coef10 =  get.coef(x=waiting, y=eruptions, lambda=10)
f.coef100 =  get.coef(x=waiting, y=eruptions, lambda=100)
f.coef1000 =  get.coef(x=waiting, y=eruptions, lambda=1000)
f.coef10000 =  get.coef(x=waiting, y=eruptions, lambda=10000)
x = seq(40, 100, length=101)
y10 = y100= y1000 = y10000 = numeric(length(x))

for(i in 1:101){
    y10[i] = nat.cubic.spline(x[i], f.coef10, waiting)
    y100[i] = nat.cubic.spline(x[i], f.coef100, waiting)
    y1000[i] = nat.cubic.spline(x[i], f.coef1000, waiting)
    y10000[i] = nat.cubic.spline(x[i], f.coef10000, waiting)
}
lines(y10~x, col="blue", lwd=2)
```

```
lines(y100~x, col="red", lwd=2)
lines(y1000~x, col="black", lwd=2)
lines(y10000~x, col="green", lwd=2)
legend(x="topleft", lwd=2, col=c("blue", "red", "black", "green"),
       legend = c("lambda = 10", "lambda = 100", "lambda = 1000", "lambda = 10000"))
```



The spline with $\lambda = 10$ seems noticeably irregular or "bumpy". I would argue that the spline with $\lambda = 10000$ over-smooths the data, over-estimating the eruption time at around 60 minutes waiting time. I would probably choose $\lambda = 100$, but arguably $\lambda = 1000$ is as good. There is often a wide range of values of λ that give similar results. (In lecture 4, we found that the optimal value of λ chosen by cross-validation was 501 – see the lecture 4 *R* script for details.)

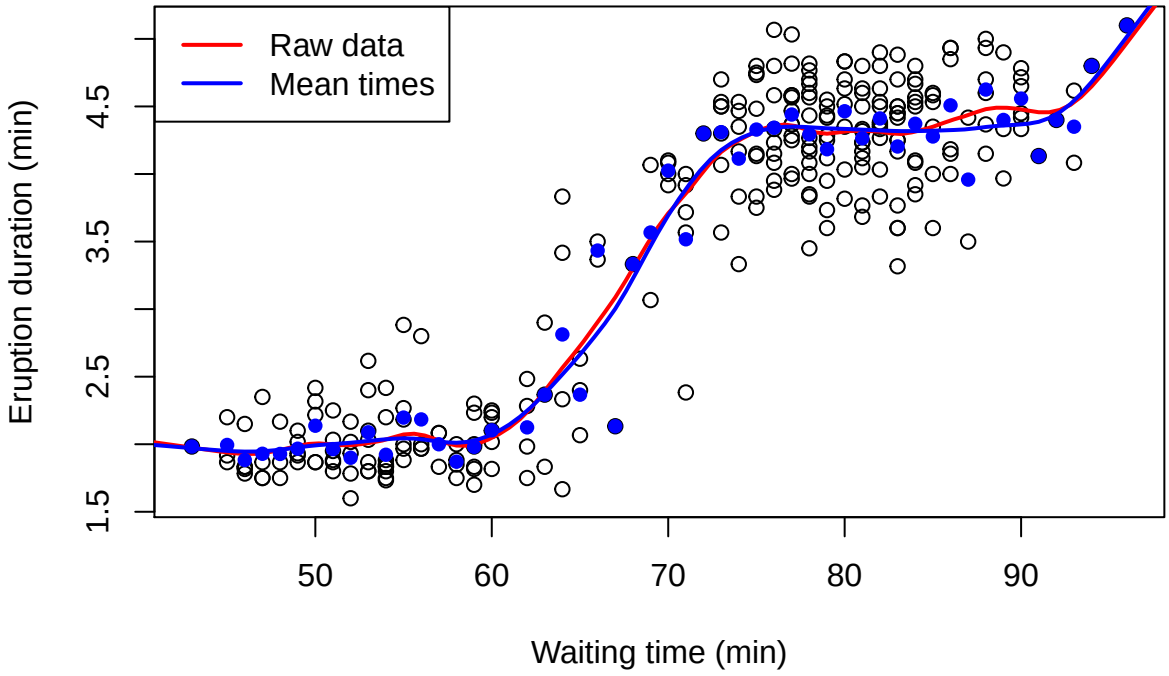
Note that many waiting times are not unique. We can still fit the spline, but does it make sense to do so? We could instead take an average mean eruption time at each knot and fit the spline to these times.

```
plot(x=waiting, y=eruptions, xlab = "Waiting time (min)",
     ylab = "Eruption duration (min)")
lines(y100~x, col="red", lwd=2)

uw = sort(unique(waiting))
ue = numeric(length(uw))
```

```
for(i in 1:length(uw))
    ue[i] = mean(eruptions[waiting==uw[i]])
points(x=uw, y=ue, pch=16, col="blue")

uf.coef = get.coef(x=uw, y=ue, lambda=100)
uy = numeric(101)
for(i in 1:101)
    uy[i] = nat.cubic.spline(x[i], uf.coef, uw)
lines(uy~x, col="blue", lwd=2)
legend(x="topleft", lwd=2, col=c("red", "blue"),
       legend=c("Raw data", "Mean times"))
```



The fit to the averaged data is slightly smoother, but not very different. Increasing λ and taking the mean eruption time are effectively both ways to further smooth the fitted spline, although they operate in slightly different ways.