# MATH5824 Solutions to Practical – Splines question

Robert G Aykroyd

5/3/23

A plot of the data is shown in Figure 1. As engine size increases, there seems to be an initial drop in the observed wear, then a gentle increase from about wear of 2.0 to 2.4, with a further decline at the higher end of the wear values.

```
engines =  read.table("https://rgaykroyd.github.io/MATH5824/Datasets/engine.txt", header =

attach(engines)

par(mgp=c(2,0.7,0))
plot(size, wear, pch=16)

fit.locations = seq(1.2,3.0,0.01)

fit1 = smooth.spline(size, wear, lambda=1e-5)
fitted1 = predict(fit1, fit.locations)
lines(fitted1, col="blue", lwd=1.5)

fit2 = smooth.spline(size, wear, lambda=1e-3)
fitted2 = predict(fit2, fit.locations)
lines(fitted2, col="red", lwd=1.5)

fit3 = smooth.spline(size, wear, lambda=1e-2)
fitted3 = predict(fit3, fit.locations)
lines(fitted3, col="black", lwd=1.5)

legend(2.6, 4.8,
       legend=expression(
         paste(lambda, "=", 10^-5 ),
```

```
    paste(lambda, "=", 10^-3 ),
    paste(lambda, "=", 10^-2 )),
  col=c("blue", "red", "black"), lty=1, lwd=2
  )
```

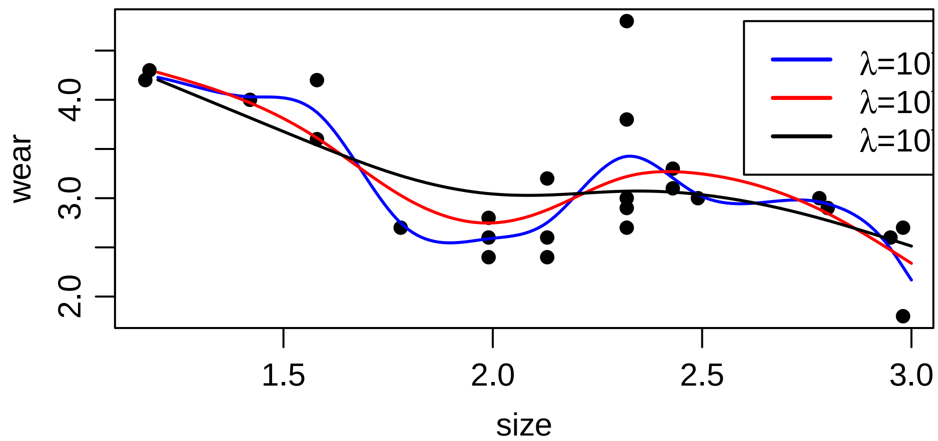


Figure 1: Engine wear index plotted against engine size, with superimposed smoothing splines.

A selection of smoothing splines with different choices of smoothing parameter $\lambda$ are also shown in Figure 1. It is clear that $\lambda = 10^{-2}$ results in too much smoothing, for larger values the result would be close to a linear fit. The behaviour of the spline with $\lambda = 10^{-5}$ seems a little irregular around engine size 1.5 and 2.5. The spline with $\lambda = 1^{-3}$ gives a good compromise between these values and seems to capture the main features of the data that we noted above.

(Although it's not required as part of these solutions, using cross-validation suggests that $\lambda = 0.0012$ is optimal.)

The behaviour of the splines for much smaller and larger $\lambda$ are shown in F@fig-exteme. As we expect, the spline becomes a linear fit as $\lambda \to \infty$. As $\lambda \to 0$, we find that the spline interpolates the mean wear at each engine size.

```
par(mgp=c(2,0.7,0))
plot(size, wear, pch=16)
```

```r
fit.locations = seq(1.12,3.0,0.01)

fit1 = smooth.spline(size, wear, lambda=1e-7)
fitted1 = predict(fit1, fit.locations)
lines(fitted1, col="blue", lwd=1.5)

fit2 = smooth.spline(size, wear, lambda=1e-1)
fitted2 = predict(fit2, fit.locations)
lines(fitted2, col="red", lwd=1.5)

legend(2.6, 4.8,
       legend=expression(
         paste(lambda, "=", 10^-7 ),
         paste(lambda, "=", 10^-1 )),
       col=c("blue", "red"), lty=1, lwd=2
       )

ymeans = unlist(lapply(split(wear, size), mean))
xunique = unique(size)
points(xunique, ymeans, pch=4, cex=2, lwd=2, col="blue")
```

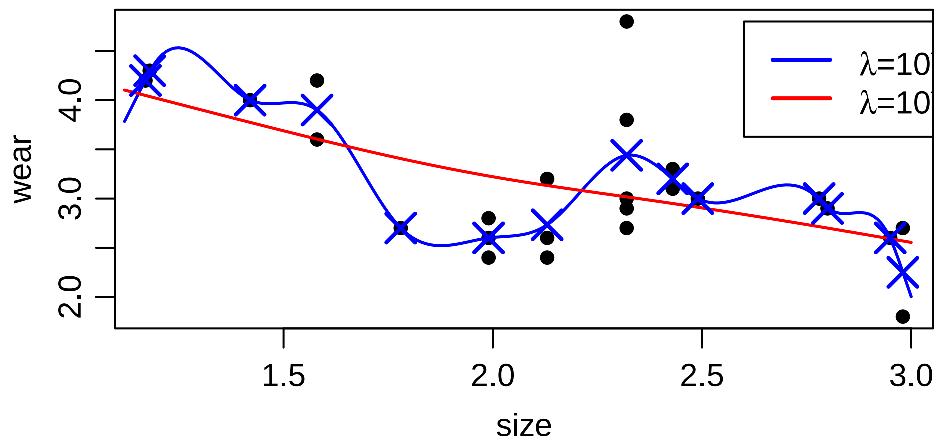


Figure 2: Engine wear index plotted against engine size, with superimposed smoothing splines. Blue crosses indicate mean wear at each engine size.

## R code Appendix

This appendix contain the **R** code used to produce these *Solutions* but they are not meant to represent the best solution and there are always other ways to organize code and to perform a valid analysis.

For Figure 1 the follow was used:

```
engines =  read.table("https://rgaykroyd.github.io/MATH5824/Datasets/engine.txt", header =

attach(engines)

par(mgp=c(2,0.7,0))
plot(size, wear, pch=16)

fit.locations = seq(1.2,3.0,0.01)

fit1 = smooth.spline(size, wear, lambda=1e-5)
fitted1 = predict(fit1, fit.locations)
```

```
lines(fitted1, col="blue", lwd=1.5)

fit2 = smooth.spline(size, wear, lambda=1e-3)
fitted2 = predict(fit2, fit.locations)
lines(fitted2, col="red", lwd=1.5)

fit3 = smooth.spline(size, wear, lambda=1e-2)
fitted3 = predict(fit3, fit.locations)
lines(fitted3, col="black", lwd=1.5)

legend(2.6, 4.8,
       legend=expression(
         paste(lambda, "=", 10^-5 ),
         paste(lambda, "=", 10^-3 ),
         paste(lambda, "=", 10^-2 )),
       col=c("blue", "red", "black"), lty=1, lwd=2
       )
```

For Figure 2 the follow was used:

```
par(mgp=c(2,0.7,0))
plot(size, wear, pch=16)

fit.locations = seq(1.12,3.0,0.01)

fit1 = smooth.spline(size, wear, lambda=1e-7)
fitted1 = predict(fit1, fit.locations)
lines(fitted1, col="blue", lwd=1.5)

fit2 = smooth.spline(size, wear, lambda=1e-1)
fitted2 = predict(fit2, fit.locations)
lines(fitted2, col="red", lwd=1.5)

legend(2.6, 4.8,
       legend=expression(
         paste(lambda, "=", 10^-7 ),
         paste(lambda, "=", 10^-1 )),
       col=c("blue", "red"), lty=1, lwd=2
       )

ymeans = unlist(lapply(split(wear, size), mean))
```

```
xunique = unique(size)
points(xunique, ymeans, pch=4, cex=2, lwd=2, col="blue")
```