MATH5824 Assessed Practical - Outline Analysis

Please note that the following only considers the extra MATH5824 question on spline regression – see the MATH3823-Practical-Solution for details of the first part.

The following is not a draft report, but is a *log* of a typical R session. Your report will then have summarized the main ideas, with the R commands moved to an appendix.

Initial data analysis

The following assumed the SID=19.

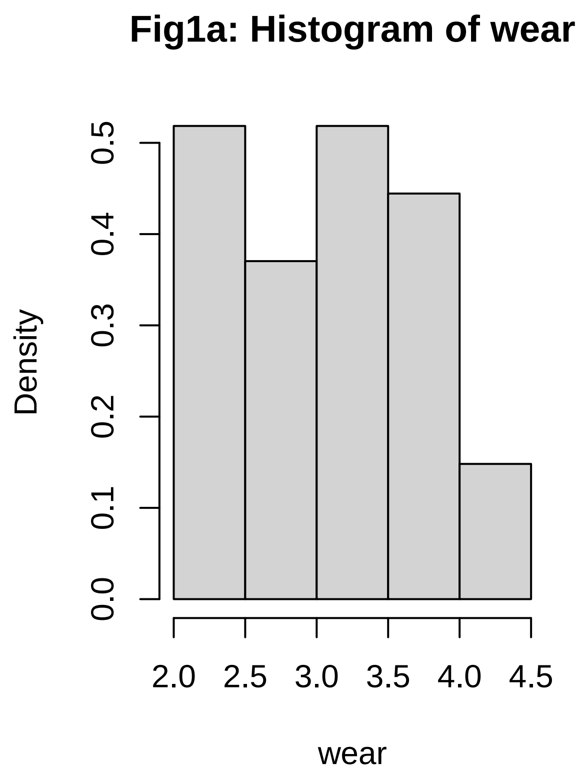
Initial exploration

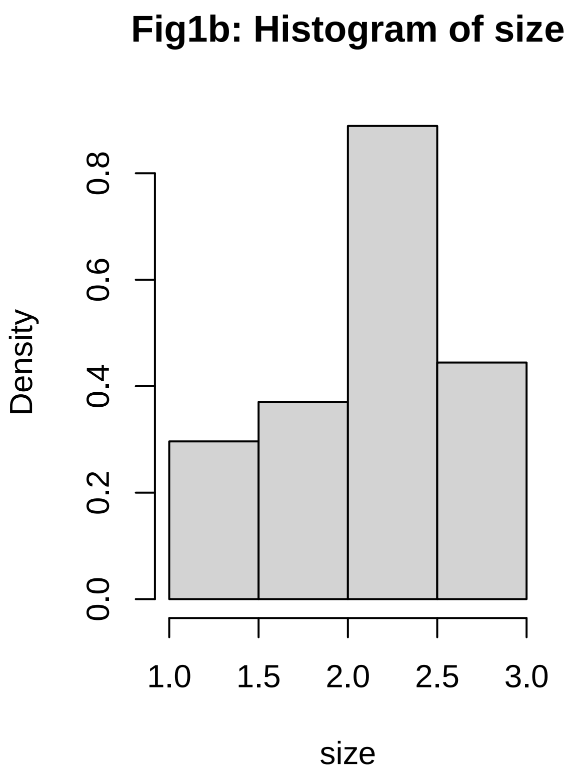
Although it might be useful to calculate the numerical summary statistics, such as here, it is more helpful to show histograms, as below in Figure 1.

```
summary(data)
```

```
##       size            wear      
##  Min.   :1.180   Min.   :2.100  
##  1st Qu.:1.885   1st Qu.:2.550  
##  Median :2.320   Median :3.300  
##  Mean   :2.216   Mean   :3.119  
##  3rd Qu.:2.490   3rd Qu.:3.600  
##  Max.   :2.980   Max.   :4.200
```

```
par(mfrow=c(1,2))
hist(wear, probability=T, main="Fig1a: Histogram of wear")
hist(size, probability=T, main="Fig1b: Histogram of size")
```





From the histograms, in Fig. 1, we see that wear has a symmetric distribution with mode between 3 and 3.5 L, and that size is negative skew, ranging from index 1 to 3 with mode between 2 and 2.5.

The scatter plot of the data set in Fig. 2 shows a general decrease in engine wear as the engine size increases. It is possible that an initial decrease from 1 to 2 L, is followed by a small increase, peaking at about 2.3 L before a further decrease. Although there is considerable variability, it appears that a linear relationship is not appropriate, and hence splines will be considered.

```r
plot(size, wear, pch=16, main="Fig 2. Engine wear against engine size",
     xlab="Engine size, L", ylab="Index of engine wear")
```

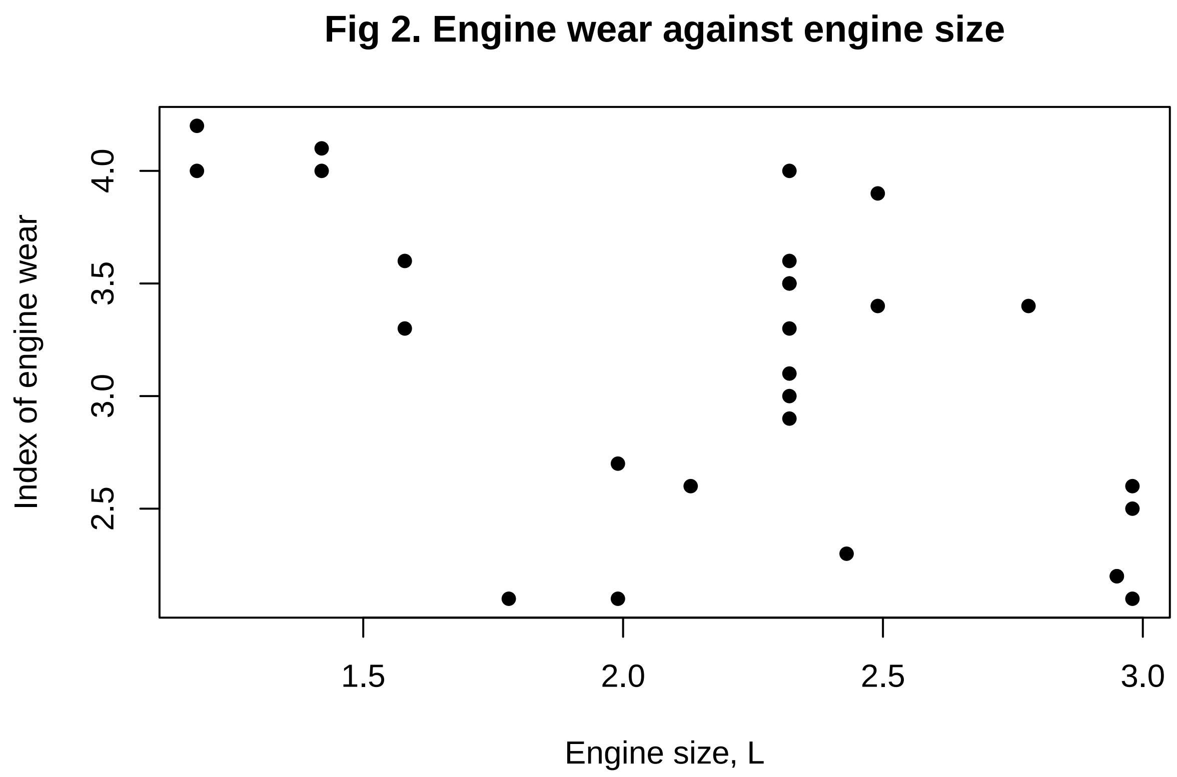
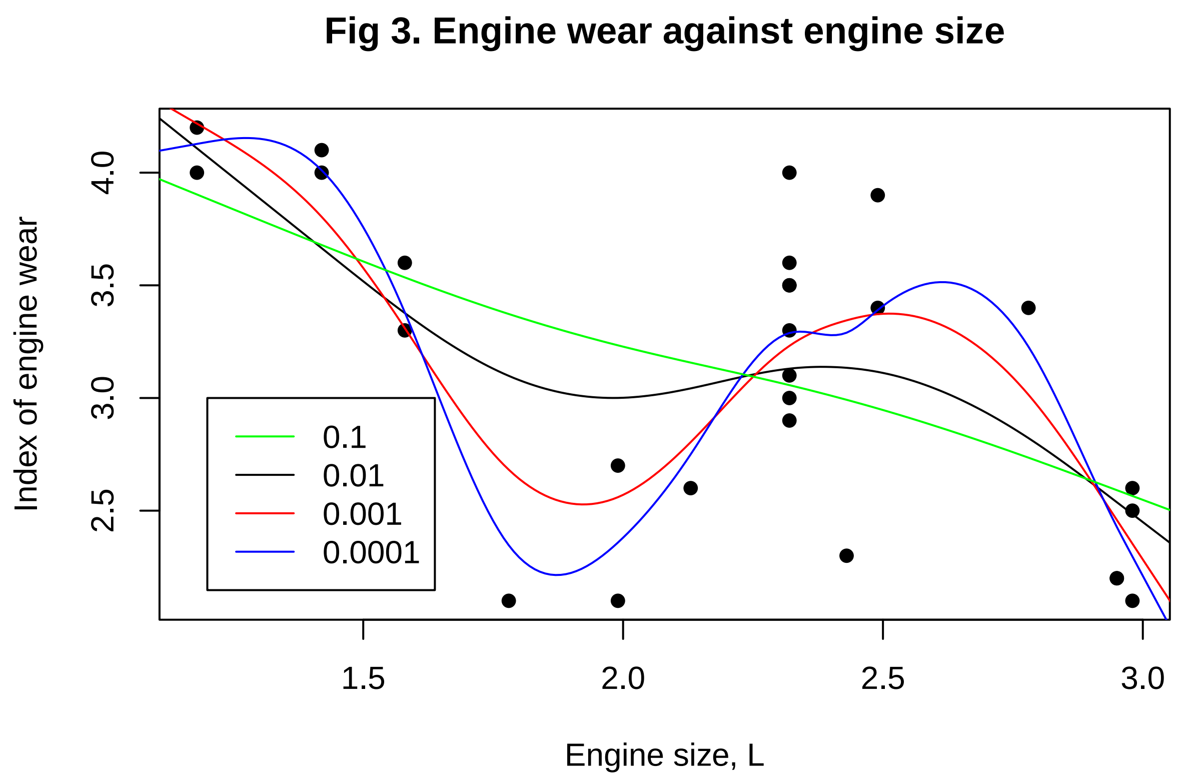


Fig 3. shows four splines regression lines with varying values of the smoothing parameter: $\lambda = \{0.1, 0.01, 0.001, 0.0001\}$

```r
plot(size, wear, pch=16, main="Fig 3. Engine wear against engine size",
     xlab="Engine size, L", ylab="Index of engine wear")

# spline with lambda=0.01
my.spline1 = smooth.spline(size, wear, lambda=0.01)

newsize = seq(1,4,by=0.01)
predicted1 = predict(my.spline1, newsize)
lines(predicted1)

# spline with lambda=0.001
my.spline2 = smooth.spline(size, wear, lambda=0.001)
predicted2 = predict(my.spline2, newsize)
lines(predicted2, col="red")

# spline with lambda=0.0001
my.spline2 = smooth.spline(size, wear, lambda=0.0001)
predicted2 = predict(my.spline2, newsize)
lines(predicted2, col="blue")

# spline with lambda=0.1
my.spline2 = smooth.spline(size, wear, lambda=0.1)
predicted2 = predict(my.spline2, newsize)
```

```r
lines(predicted2, col="green")


legend(1.2,3, lty=1, col=c("green",1,"red","blue"), legend=c("0.1","0.01","0.001","0.0001"))
```



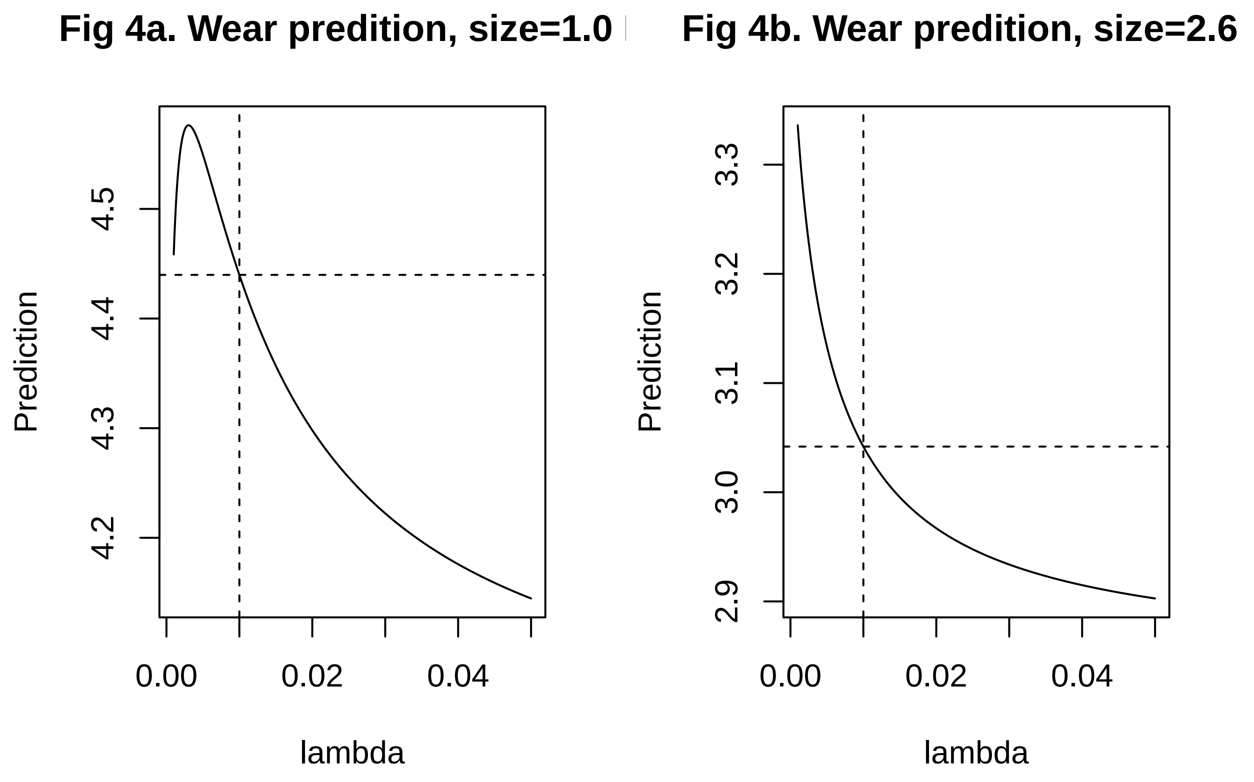
For small values of the smoothing parameter, the spline is very “wiggly” and tends to follow the data points and in the limit as $\lambda \to 0$ then the spline will interpolate the data. The spline with $\lambda = 0.01$ appears to be a good fit to the data as it shows the general trend without over-fitting the data. As λ further increases the spline no longer captures the trend in the data, and as $\lambda \to \infty$ the spline tends to a straight line – which coincides with the usual linear regression.

Of these, the best value is $\lambda = 0.01$ with any value between about 0.005 to 0.05 likely to result in a reasonable fit. If we believe the dip ar about 2.0 L, then the value of $\lambda = 0.001$ is, however, better.

Using the value $\lambda = 0.01$ the predicted wear for a car with a 1.0 L engine is 4.44 and for a 2.6 L engine it is 3.04.

Although it is acceptable to try a few values of the smoothing parameter around $\lambda = 0.01$, to see the sensitivity to the choice of λ, here it has been automated:

```r
vlambda = seq(0.001,0.05,0.0001)
nlambda = length(vlambda)

xval1 = NULL
xval2 = NULL

for (i in 1:nlambda){
    my.spline2 = smooth.spline(size, wear, lambda=vlambda[i])
    xval1[i] = predict(my.spline2, 1.0)$y
    xval2[i] = predict(my.spline2, 2.6)$y
}

par(mfrow=c(1,2))
plot(vlambda, xval1, type='l', ylab="Prediction", xlab="lambda",
     main="Fig 4a. Wear predition, size=1.0 L")
abline(v=0.01, lty=2); abline(h=predict(my.spline1, 1.0)$y, lty=2)

plot(vlambda, xval2, type='l', ylab="Prediction", xlab="lambda",
     main="Fig 4b. Wear predition, size=2.6 L")
abline(v=0.01, lty=2); abline(h=predict(my.spline1, 2.6)$y, lty=2)
```

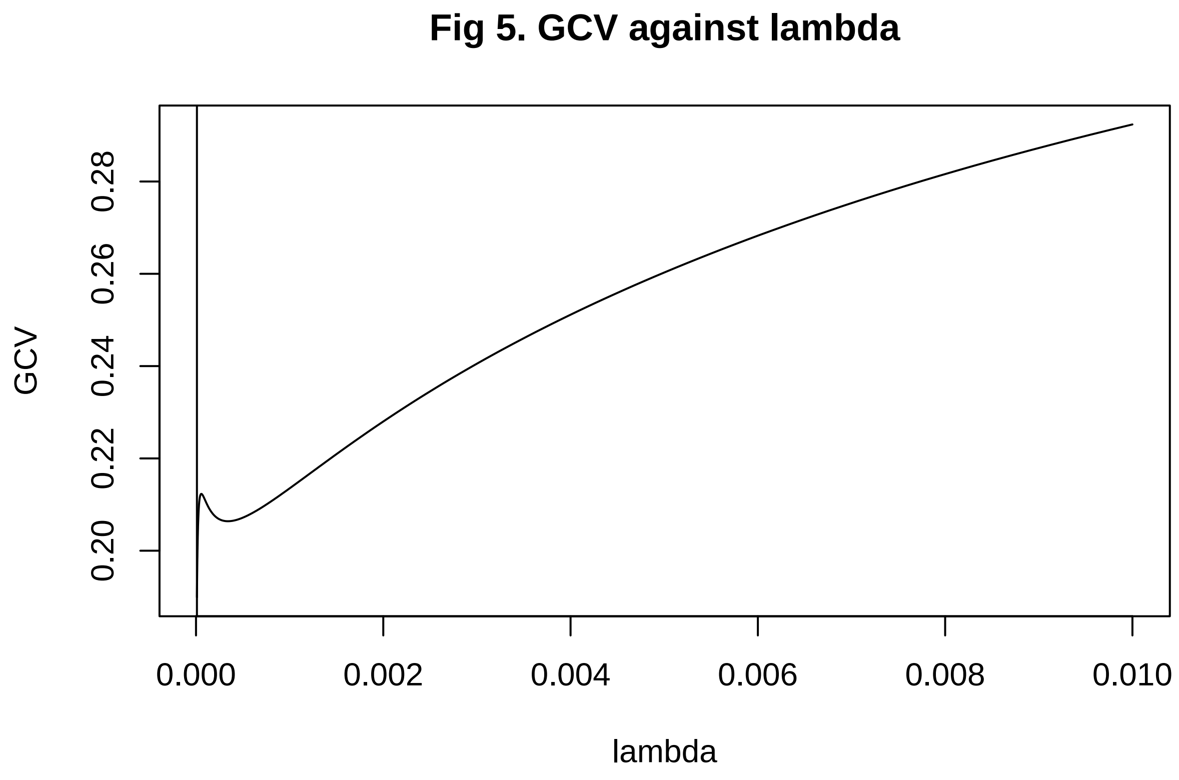


From the multiple predictions with varying λ, it is clear that if $\lambda < 0.01$ both predictions indicate greater wear, and lower wear if $\lambda > 0.01$. Over a moderate range of λ values, however, the change is not substantial.

One automatic method for choosing λ is generalized cross-validation (ordinary cross-validation is also reasonable). This method aims to reduce over-fitting by fitting and assessing the fit on different sub-sets of the data. In particular, one data point is left out and its value predicted using the spline fitted to the rest of the data. This is done removing each data point in turn and the squares prediction errors are then average. This process is repeated for different values of λ. The best λ is then taken as the value which minimizes the cross-validation criterion.

It is perfectly acceptable to find the minimum GCV estimate of λ by trial and error, the following, however, automates the process.

```
vlambda = 10^(seq(-5,-2,0.01))
nlambda = length(vlambda)

xval = NULL

for (i in 1:nlambda){
  my.spline1 = smooth.spline(size, wear, lambda=vlambda[i])
  xval[i] = my.spline1$cv.crit
}

lambda.gcv = vlambda[which.min(xval)]

plot(vlambda, xval, type='l', xlab="lambda", ylab="GCV", main="Fig 5. GCV against lambda")

abline(v=lambda.gcv)
```



The minimum of this is at $\hat{\lambda}_{GCV} = 10^{-5}$ which can be obtained directly from the r command

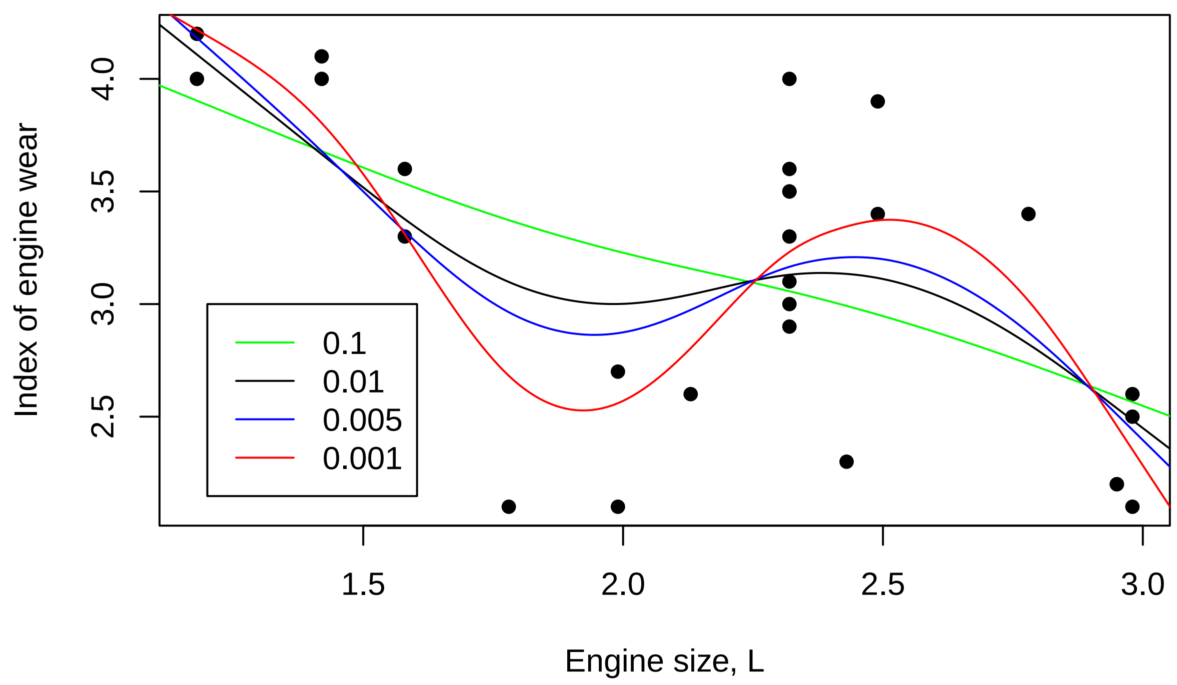
```
my.spline1 = smooth.spline(size, wear)
my.spline1$lambda
```

```
## [1] 1.362381e-06
```

Finding a monotonic spline can be done by trial and error, for example using $\lambda = \{0.1, 0.01, 0.001, 0.0001\}$

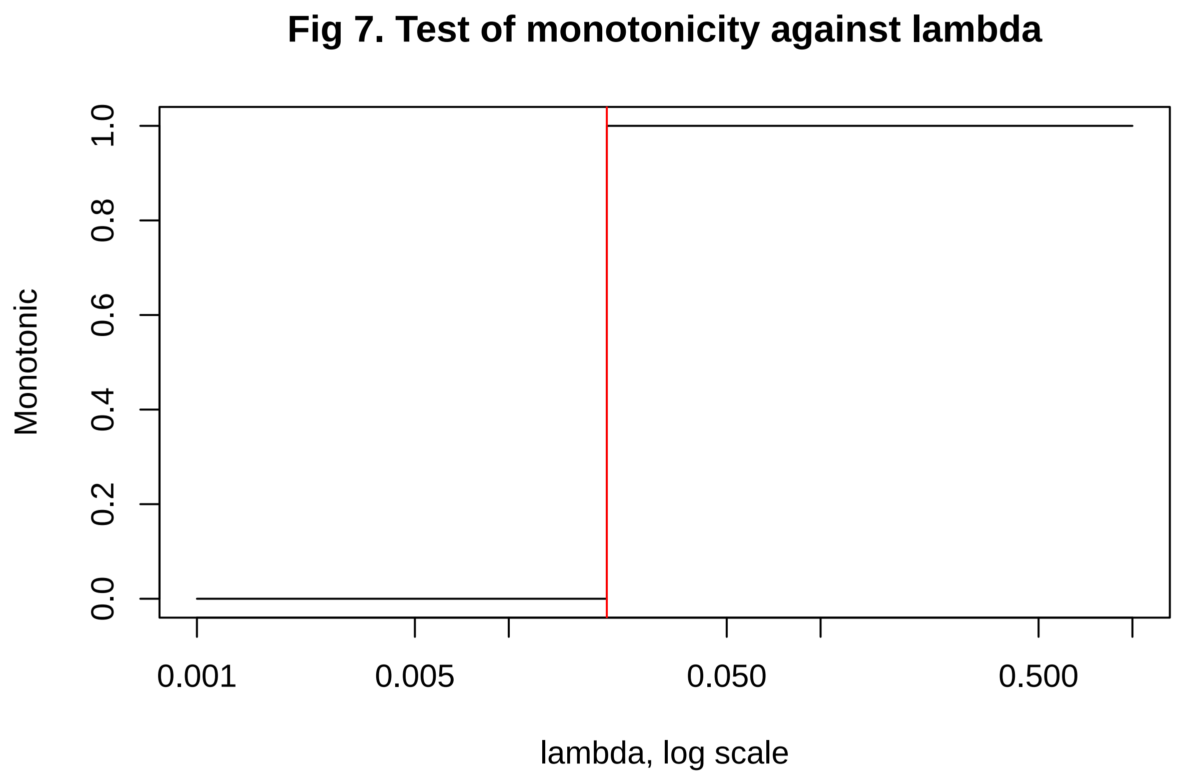
```r
plot(size, wear, pch=16, main="Fig 6. Engine wear against engine size",
     xlab="Engine size, L", ylab="Index of engine wear")

# spline with lambda=0.1
my.spline2 = smooth.spline(size, wear, lambda=0.1)
predicted2 = predict(my.spline2, newsize)
lines(predicted2, col="green")

# spline with lambda=0.01
my.spline1 = smooth.spline(size, wear, lambda=0.01)
newsize = seq(1,4,by=0.01)
predicted1 = predict(my.spline1, newsize)
lines(predicted1, col="black")

# spline with lambda=0.005
my.spline2 = smooth.spline(size, wear, lambda=0.005)
predicted2 = predict(my.spline2, newsize)
lines(predicted2, col="blue")

# spline with lambda=0.001
my.spline2 = smooth.spline(size, wear, lambda=0.001)
predicted2 = predict(my.spline2, newsize)
lines(predicted2, col="red")


legend(1.2,3, lty=1, col=c("green",1, "blue","red"), legend=c("0.1","0.01","0.005", "0.001"))
```

Fig 6. Engine wear against engine size



or automated as follows:

```
vlambda = 10^seq(-3, 0, by=0.0001)
nlambda = length(vlambda)

xval = NULL

for (i in 1:nlambda){
  my.spline1 = smooth.spline(size, wear, lambda=vlambda[i])
      tmp=diff(my.spline1$y)
      xval[i] = abs(sum(tmp))==sum(abs(tmp))
}

lambda.mono = vlambda[xval][1]

plot(vlambda, xval, type='l', xlab="lambda, log scale", ylab="Monotonic", main="Fig 7. Test of monotoni

abline(v=lambda.mono, col="red")
```



This leads to a value of $\hat{\lambda}_{mono} = 0.02063$.

End of MATH5824 Assessed Practical - Outline Analysis