Splines

Sergi Carol Laura Cebollero Alex Rodriguez

December 17, 2018

Introduction

The aim of this lab is to understand the spline smoothing effects on regression methods, the lab will consist of two different tasks, the first one will combine the use of a **b-spline** basis with a liniar model in order to compute the optimal fitting for our data. The second task will use the method **smooth spline** in order to calculate the optimal fitting.

Exercise 1

We are asked to combine the functions bs and lm to estimate `log(Fat)` as a function of `abs.850` with a cubic spline (thus, k = 3).

So we first check the summary of the data and apply a logarithmic scale to `Fat` and sort the data according to `abs.850`:

```
summary(meat)
```

```
##       Fat           abs.850         abs.957     
##  Min.   : 0.90   Min.   :2.066   Min.   :2.572  
##  1st Qu.: 7.30   1st Qu.:2.512   1st Qu.:3.083  
##  Median :14.00   Median :2.754   Median :3.382  
##  Mean   :18.14   Mean   :2.809   Mean   :3.462  
##  3rd Qu.:28.00   3rd Qu.:3.006   3rd Qu.:3.714  
##  Max.   :49.10   Max.   :4.237   Max.   :5.299  
```

```
y = log(meat$Fat)
x = meat$abs.850

# sort data
sx = sort(x, index.return=T)
x = sx$x
y = y[sx$ix]
```

The next step is to establish the number of degrees to use, which we have established to be 3, for we are asked to perform the nonparametric regression with a cubic spline.

To obtain the best number of knots to use with the spline, we have implemented a 10-fold cross-validation to determine it and a function that computes the prediction of values to check how well our model fits the data:

```
computePredictions <- function(x, n.knots, k, y, x.val) {
  my.knots <- quantile(x$abs,seq(0,1,length=n.knots))
  l = length(my.knots)
  inner.knots <- my.knots[-c(1,l)]

  df = n.knots + k + 1
  data <- cbind(x,y)
  basis <- lm(y~bs(abs, knots=inner.knots,intercept=T,degree=k, df=df), data=data)
  res <- predict(basis, x.val)
  return(res)
```

```r
}


f.10.CV.inner.knots <- function(x, y, k, n.knots.range) {
  n <- length(x)
  sample <- sample(rep(1:10, length=n), n)
  k.fold.df = data.frame()
  for (ki in 1:10){
    X.train <- data.frame(abs=x[which(sample!=ki)])
    X.val <- data.frame(abs=x[which(sample==ki)])
    Y.train <- y[which(sample!=ki)]
    Y.val <- y[which(sample==ki)]
    r.sq.array = c()
    possible.knots = seq(n.knots.range[1], n.knots.range[2])
    for(n.knots in possible.knots){
      fitted.vals = computePredictions(X.train, n.knots, k, Y.train, X.val)
      (r.sq = sum(((Y.val - fitted.vals)^2)/length(Y.val)))
      r.sq.array = append(r.sq.array, r.sq)
    }

    minRsq = which(min(r.sq.array) == r.sq.array); minRsq = minRsq[1]
    k.fold.df = rbind(k.fold.df, c(minRsq, r.sq.array[minRsq]))
  }
  print(k.fold.df)
  minIdx = which(min(k.fold.df[, 2]) == k.fold.df[,2])[1]
  return(k.fold.df[minIdx, 1])
}
```

Once we have the function, we use it to compute the best number of nodes. We establish a range of knots to test from 1 to 20.

```r
k = 3
n.knots.optim = f.10.CV.inner.knots(x, y, k, c(1, 20))
```

```
##    X11 X0.335301540694455
## 1   11          0.3353015
## 2    3          0.4770494
## 3   18          0.3802682
## 4   16          0.3317849
## 5   19          0.3433015
## 6    3          0.7728477
## 7    8          0.5275469
## 8    9          0.4714557
## 9   19          0.4887789
## 10  10          0.3809449
```

```r
cat("\n Number of optimum knots: ", n.knots.optim, "\n")
```

```
## 
##  Number of optimum knots:  16
```

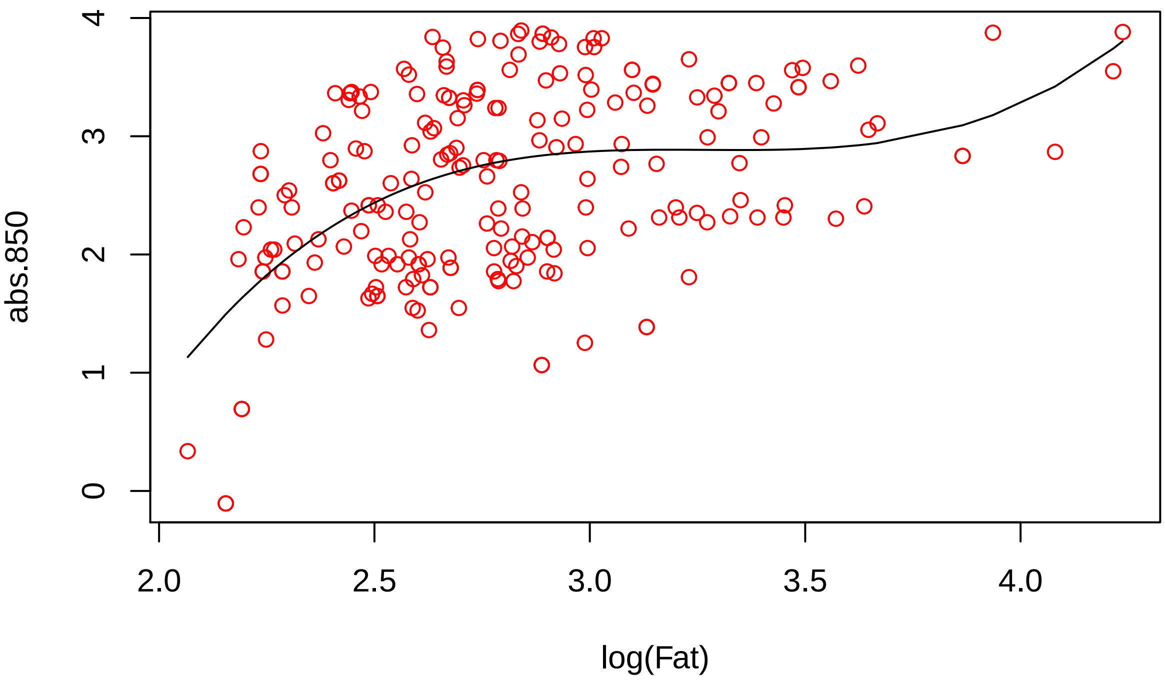
We can see that the best number of knots for our 10-fold CV is 16.

Now we can proceed onto use this number of knots to create a cubic spline to model our data:

```r
degrees = n.knots.optim + k + 1
x<- data.frame(abs=x)
```

```
data <- cbind(y,x)
optim.spline <- lm(y~bs(abs, knots=n.knots.optim,intercept=T,degree=k, df=df), data=data)


plot(x$abs,y,col=2,xlab="log(Fat)",ylab="abs.850")
lines(x$abs,optim.spline$fitted.values)
```



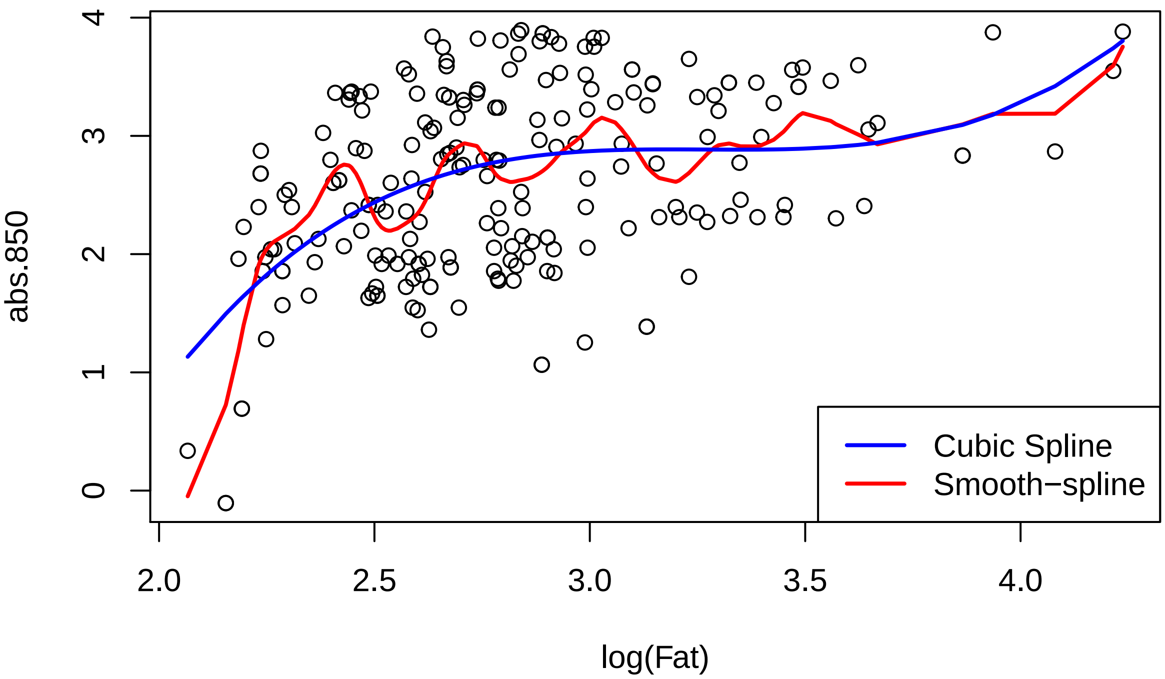
We can see that with 16 knots, the spline models the data well enough.

Exercise 2

In this task we will use the method smooth spline with the previous computed number of effective parameters. We will then plot the results of both this method and the previous used method in order to compare them.

```
degrees = n.knots.optim + k + 1

m1  <- smooth.spline(x$abs, y, df = degrees)
plot(x$abs,y, xlab="log(Fat)",ylab="abs.850")
lines(m1, col="red", lwd=2)
lines(x$abs,optim.spline$fitted.values, col="blue", lwd=2)
legend("bottomright", legend = c("Cubic Spline", "Smooth-spline"),
           lty = 1, lwd = 2,col = c("blue", "red"))
```



We can see how the Smooth spline is not, in fact, as smooth as our cubic spline.

Thus, we seem to see that our implementation to find a fitting cubic spline fits the data more smoothly and without overfitting, whereas the Smooth-spline function seems to overfit the data and is more biased based on the present data.