Lab 1: Splines

Please log into the computer using the credentials given to you in the delegate pack. Please note that each account is specific to one computer.

In this lab you will need a data object which you can load using the command

```
load(url("http://www.stats.gla.ac.uk/~levers/aptslab1.RData"))
```

Data

In this lab you can work with any of the following three datasets:

Divorces in the US

This data set (called `divorces`) contains the number of divorces per 10,000 women for most of the 20th century. it has the following columns:

`year`	Calendar year
`divorces`	Number of divorce per 1,000 women (aged 15+) in that year *(response)*

Radiocarbon dating.

This is the data (called `radiocarbon`) used in the lectures

`cal.age`	True calendar age
`rc.age`	Age predicted from the radiocarbon dating process *(response)*

Great barrier reef.

This is a univariate version of the data (called `gbr`) used in the lectures. Only two columns are retained:

`longitude`	Longitude
`score1`	Principal component score summarising the catch *(response)*

Once you have loaded the above `.RData` *file all datasets should be available in your workspace.*

Tasks

1. Explore the data set you have chosen graphically.
2. Launch the function `pspline.cartoon` with the name of your dataset as argument i.e.

   Divorces in the US. `pspline.cartoon(divorces)`

   Radiocarbon dating. `pspline.cartoon(radiocarbon)`

   Great barrier reef. `pspline.cartoon(gbr)`

The function fits a P-spline model to the data and shows the fitted model together with the basis function.

Experiment with the degree of the spline, the number of knots and the smoothing parameter λ.

3. (a) Using the `lm` function in R, fit a polynomial regression model of appropriate degree to the data. Use the `predict` (or `fitted`) method to compute the predicted values and plot them.

   If necessary, change the degree of the polynomial.

   Hint: The formula `y~x+I(x^2)` *or* `y~poly(x,2)` *fits a quadratic regression model in R.*

   (b) Construct the design matrix for polynomial regression and compute the regression coefficients and fitted values "by hand" using

$$\mathbf{B} = \begin{pmatrix} 1 & x_1 & \dots & x_1^r \\ \vdots & \vdots & \ddots & \vdots \\ 1 & x_n & \dots & x_n^r \end{pmatrix}$$
$$\hat{\boldsymbol{\beta}} = (\mathbf{B}^\top \mathbf{B})^{-1} \mathbf{B}^\top \mathbf{y}$$
$$\hat{\mathbf{y}} = \mathbf{B}\hat{\boldsymbol{\beta}}$$

   For the remaining tasks (expect for the last) you only need to modify the line of code which creates the matrix of basis function **B**.

4. Now fit a B-spline-based regression model to the data. You can create a B-spline basis using the function `bbase` (supplied in `aptslab1.RData`)

```
R 1 B <- bbase(x, deg=3, n.knots=10)
```

   Adjust the parameter `n.knots` if necessary. Does is yield better results than the polynomial model?

5. (harder) Now fit a spline model using the truncated power series basis. Remember the design matrix is

$$\mathbf{B} = \begin{pmatrix} 1 & x_1 & \dots & x_1^{r-1} & (x_1-\kappa_1)_+^r & \dots & (x_1-\kappa_{l-1})_+^r \\ \vdots & \vdots & \ddots & \vdots & \vdots & \ddots & \vdots \\ 1 & x_n & \dots & x_n^{r-1} & (x_n-\kappa_1)_+^r & \dots & (x_n-\kappa_{l-1})_+^r \end{pmatrix}$$

   for a truncated power series basis of degree r with equally-spaced knots in $\kappa_1, \dots, \kappa_l$.

   Compare your results to the results you have obtained in task 4.

   Hint: You can create sequence of l equally spaced knots using the command `knots <- seq(min(x), max(x), len=l)`.

6. Finally we fit a P-spline model. Remember that the penalised least squares estimate is

$$\hat{\boldsymbol{\beta}} = (\mathbf{B}^\top \mathbf{B} + \lambda \mathbf{D}^\top \mathbf{D})^{-1} \mathbf{B}^\top \mathbf{y}.$$

   Implement this formula and then plot the fitted values.

   Hint: You can create the differencing matrix **D** *of order d using the R code* `D <- diff(diag(ncol(B)), diff=d)` *(assuming you have called the design matrix* `B`*).*

7. (bonus task) Inverting the matrices $\mathbf{B}^\top \mathbf{B}$ and $\mathbf{B}^\top \mathbf{B} + \lambda \mathbf{D}^\top \mathbf{D}$ is numerically less stable than using a QR decomposition. Use a QR decomposition to compute $\hat{\boldsymbol{\beta}}$ in the above tasks.

   Hint: You might find the functions `qr` *and* `qr.coef` *useful.*

Lab 1: Splines – Model answers

```
load(url("http://www.stats.gla.ac.uk/~levers/aptslab1.RData"))
# We use the divorces data as an example
data <- data.frame(x=divorces[,1], y=divorces[,2])
attach(data)
```

1. Something simple like

   ```
   plot(data$x, data$y)
   ```

   should be fine.

2. —

3. Using the built-in functions …

   ```
   model <- lm(y~poly(x,10), data=data)
   yhat <- predict(model)
   plot(x, y)
   lines(x, yhat)
   ```

   Coding from first principles …

   ```
   r <- 4
   B <- outer(x, 0:r, "^")
   beta <- solve(crossprod(B), t(B)%*%y)
   yhat <- B%*%beta
   ```

   Plotting code is same as above. Higher power might not work because of numerical instability. The code using `lm` and `poly` is more stable because `lm` uses a QR decomposition, which is numerically more stable, and because `poly` creates orthogonal polynomials.

4. Simply define `B` using

   ```
   B <- bbase(x, deg=3, n.knots=10)
   ```

5. We need to create the basis by hand (which we will do in a function called `tbase`.

   ```
   tbase <- function(x, xl = min(x), xr = max(x), n.knots = 10, deg = 3) {
     nseg <- n.knots - 1
     dx <- (xr - xl) / nseg
     knots <- seq(xl, xr,  by = dx)
     B <- cbind(outer(x-xl,0:(deg-1),"^"),
                outer(x,knots[-length(knots)],function(x,y) pmax(x-y,0))^deg)
     B
   }
   B  <- tbase(x, n.knots=10, deg=3)
   ```

   Both bases are equivalent, so the fitted values are exactly the same.

6. After creating `B` we insert

   ```
   D <- diff(diag(ncol(B)), diff=2)
   lambda <- 1 # or whatever value you like
   beta <-  solve(crossprod(B)+lambda*crossprod(D), t(B)%*%y)
   yhat <- B%*%beta
   ```

7. For the unpenalised models use

```
R 29 beta <- qr.coef(qr(B), y)
```

For P-splines use

```
R 30 beta <- qr.coef(qr(rbind(B, sqrt(lambda)*D)), c(y, numeric(nrow(D))))
```