LECTURE 7

Flexible regression
Some graphical examples of the use of flexible regression:
The following simple data set will be used to demonstrate a simple spline regression.
First up we fit a polynomial regression model to the data:

ie,

\[ \hat{y} = a + bx^2 + \ldots + x^6 \]
A graph labeled "polynomial" shows a plot with the x-axis labeled "x" and the y-axis labeled "y". The x-axis ranges from 5 to 15, and the y-axis ranges from 10 to 50. The data points form a trend that appears to be polynomial in nature.
Next, we try a B–spline.

This is a smooth function of $x$, with some properties that can make it superior to a polynomial in some ways.
R Code:

```r
xydf <- read.table("ryan.txt",header=T)
attach(xydf)
plot(y~x)

library(splines)

#POLYNOMIAL

lm1 <- lm(y ~ poly(x,6),data=xydf)
predvals <- predict(lm1,se=T)
xydf$fit1 <- predvals$fit

#B-SPLINE

lm3 <- lm(y ~ bs(x,df=6),data=xydf)
```
predvals3 <- predict(lm3, se=T)
xydf$fit3 <- predvals3$fit
plot(xydf$x, xydf$y, xlab="x", ylab="y", las=1, main="polynomial")
lines(xydf$x, xydf$fit1, lty=2)

plot(xydf$x, xydf$y, xlab="x", ylab="y", las=1, main="B spline")
lines(xydf$x, xydf$fit3)

print(anova(lm1))
print(anova(lm3))
A B–spline is a piecewise polynomial approximation to the data whose definition changes with key points over the range of the predictor variable. These key points are called knots, and are fixed for B–splines.
In R, simply use `bs` from the library `splines`, and set the `df` option rather than `knots`, as this an easier way to control the type of spline fitted. The option `df` chooses a B–spline equivalent to the corresponding polynomial of order `df`. 
The AOV tables for the two models:

```r
> print(anova(lm1))
Analysis of Variance Table

Response: y

<table>
<thead>
<tr>
<th>Df</th>
<th>Sum Sq</th>
<th>Mean Sq</th>
<th>F value</th>
<th>Pr(&gt;F)</th>
</tr>
</thead>
<tbody>
<tr>
<td>poly(x, 6)</td>
<td>6</td>
<td>2507.85</td>
<td>417.97</td>
<td>505.46</td>
</tr>
<tr>
<td>Residuals</td>
<td>11</td>
<td>9.10</td>
<td>0.83</td>
<td></td>
</tr>
</tbody>
</table>
```

```r
> print(anova(lm3))
Analysis of Variance Table

Response: y

<table>
<thead>
<tr>
<th>Df</th>
<th>Sum Sq</th>
<th>Mean Sq</th>
<th>F value</th>
<th>Pr(&gt;F)</th>
</tr>
</thead>
<tbody>
<tr>
<td>bs(x, df = 6)</td>
<td>6</td>
<td>2508.04</td>
<td>418.01</td>
<td>516.57</td>
</tr>
<tr>
<td>Residuals</td>
<td>11</td>
<td>8.90</td>
<td>0.81</td>
<td></td>
</tr>
</tbody>
</table>
```

The two models are very similar, in terms of explanatory power.
The fuel data example, where the simple linear model was inadequate:
The spline equation:

\[ y = \sum_{\nu=1}^{v} s_{\nu}(x_{\nu}) + \epsilon, \quad \epsilon \sim N(0, \sigma^2) \]  

(3)
The polynomial equation:

\[ y_{ij} = \beta_0 + \beta_1 x_{ij} + \beta_2 x_{ij}^2 + \epsilon_{ij} \epsilon_{ij} \sim N(0, \sigma^2) \]  \hspace{1cm} (4)

\begin{verbatim}
lm2 <- lm(Mileage ~ Weight + I(Weight^2), data=fuel)
Df Sum Sq Mean Sq F value Pr(>F)
Weight  1   973.75  973.75 173.91  0.0000
I(Weight^2)  1    61.69  61.69  11.02  0.0016
Residuals 57  319.14    5.60
\end{verbatim}
library(splines)
lm3 <- lm(Mileage ~ bs(Weight, df=6), data=fuel)

<table>
<thead>
<tr>
<th></th>
<th>Df</th>
<th>Sum Sq</th>
<th>Mean Sq</th>
<th>F value</th>
<th>Pr(&gt;F)</th>
</tr>
</thead>
<tbody>
<tr>
<td>bs(Weight, df = 6)</td>
<td>6</td>
<td>1040.71</td>
<td>173.45</td>
<td>29.29</td>
<td>0.0000</td>
</tr>
<tr>
<td>Residuals</td>
<td>53</td>
<td>313.88</td>
<td>5.92</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
The dotted line is the linear model fitted to the data:

**Quadratic**

**B-spline**
So we obtain similar results for both methods, as for the simple example.

For this data set and the simple data set, it is impossible to see any difference between the ordinary polynomial and the B–spline, so what is the advantage of the B–spline?
The potential advantage of the B–spline is best shown by an example.

**Example**

A medical trial was set up to determine the dependence of the level of *serum C–peptide* on various factors one of which was *age* of the patient. The data are shown in the graph:
A 12 degree polynomial exhibits instability at the extremes of age and near the region age=5.
polynomial

C peptide vs age

The figure shows a scatter plot of C peptide levels against age, with a polynomial trend line indicating a non-linear relationship.
The corresponding B-spline on the same df, provides a much smoother estimate over the entire range of age.
This smoother estimation is the singular advantage of splines over polynomials.

> print(anova(lm1))
Analysis of Variance Table

Response: Cpeptide

<table>
<thead>
<tr>
<th></th>
<th>Df</th>
<th>Sum Sq</th>
<th>Mean Sq</th>
<th>F value</th>
<th>Pr(&gt;F)</th>
</tr>
</thead>
</table>
poly(age, 12)| 12 | 9.1621 | 0.7635  | 1.8114  | 0.09198 |
| Residuals  | 30 | 12.6449| 0.4215  |         |        |

> print(anova(lm3))
Analysis of Variance Table

Response: Cpeptide

<table>
<thead>
<tr>
<th></th>
<th>Df</th>
<th>Sum Sq</th>
<th>Mean Sq</th>
<th>F value</th>
<th>Pr(&gt;F)</th>
</tr>
</thead>
</table>
bs(age, df = 12) | 12 | 8.5094 | 0.7091  | 1.5998  | 0.1448 |
| Residuals  | 30 | 13.2976| 0.4433  |         |        |

The polynomial model as expected gives a marginally better MSE.
R Code:

```r
handtddf <- read.table("handt.txt",header=T)
attach(handtddf)
plot(Cpeptide~age)
print(cbind(handtddf$age,handtddf$Cpeptide))
oh <- order(handtddf$age)
handtddf <- handtddf[oh,]

library(splines)

lm1 <- lm(Cpeptide ~ poly(age,12),data=handtddf)
predvals <- predict(lm1,se=T)
handtddf$fit1 <- predvals$fit
print(handtddf$fit1)
```
lm3 <- lm(Cpeptide ~ bs(age, df=12), data=handtdf)
predvals3 <- predict(lm3, se=T)
handtdf$fit3 <- predvals3$fit
print(handtdf$fit3)

plot(handtdf$age, handtdf$Cpeptide, xlab="age", ylab="Cpeptide", las=1, main="polynomial")
points(handtdf$age, handtdf$fit1, col="blue", type="l")

plot(handtdf$age, handtdf$Cpeptide, xlab="age", ylab="Cpeptide", las=1, main="B spline")
points(handtdf$age, handtdf$fit3, col="red", type="l", lty=2)

print(anova(lm1))
print(anova(lm3))
Polynomials give a *global* model for the data, while splines give a *local* fit. Thus local features may be better captured by splines than polynomials, but the polynomial gives an expression for the model that can be interpreted over the entire range of the data. In short, the choice depends on the motivation behind modelling the data, ie, simple comparison of treatments versus explanatory modelling over the entire range of the data.

The examples given at the start of this lecture are examples of the former, ie, where a comparison of treatments is the main concern. The actual form of the response to treatments is of secondary interest.
Example
The changes in body temperature of deer over a year are presented as Metabolisable energy fortnightly. The interval where temperature peaks was of interest. A polynomial of degree 6 and a B–spline (df=6) were fitted to data for a single animal.
The estimate of the peak level by splines in greater than 220, and less than 220 using the polynomial. Thus the CI for the estimate via the B–spline is narrower than for the polynomial, using the same confidence band.
The global vs local nature of the two methods can be seen by examination of the design matrices for polynomials vs splines. The full layout is given in Table 7.1 of the Notes on page 114. These design matrices can be summarised diagrammatically by

<table>
<thead>
<tr>
<th>Polynomial</th>
<th>B–Spline</th>
</tr>
</thead>
<tbody>
<tr>
<td>1 x x x x</td>
<td>x 0 0</td>
</tr>
<tr>
<td>1 x x x x</td>
<td>x x 0</td>
</tr>
<tr>
<td>1 x x x x</td>
<td>x x x</td>
</tr>
<tr>
<td>1 x x x x</td>
<td>0 x x</td>
</tr>
<tr>
<td>1 x x x x</td>
<td>0 0 x</td>
</tr>
<tr>
<td>1 x x x x</td>
<td>0 0 0</td>
</tr>
</tbody>
</table>
Both design matrices are of the same dimension, but the local nature of the B–Spline can be seen in the coverage of the columns of the design matrix. While there is overlap between each predictor (B–spline), each B–spline covers a different range of the x–axis (time in weeks). By contrast, each column of the design matrix for the polynomial model covers the entire range of the x–axis.
library(splines)
bs.mod <- lm(ME ~ bs(Age, knots=c(50,60,70)), data=tag207, x=T)
predvals <- predict(bs.mod, se=T)
tag207$bs.fit <- predvals$fit
tag207$bs.ul <- predvals$fit + 2*predvals$se.fit
tag207$bs.ll <- predvals$fit - 2*predvals$se.fit
The difference in the predictor base for the two models can be shown graphically. Note that we have six curves in both cases. The polynomials are defined over the entire time axis, while the B–splines, while overlapping, only each cover a section of the time axis, thus showing the local nature of the approximation.
polynomial

B = spline

weeks

polynomials

B - spline bases
Cubic Splines

The diagram shows the overlap for the base functions that are used for the fit over a section between two knots.
\[ y_i = s(x_i) + \epsilon, \quad \epsilon \sim N(0, \sigma^2) \]

\[ = \alpha_0 + \sum_{j=1}^{m} \alpha_j |x_i - z_j|^3 + \epsilon \]

The degrees of freedom depend on the number of knot points.

The penalty function for splines is the roughness defined by

\[ \int [s''(x)]^2 \, dx = \alpha^T (X^T X)^{-1} \alpha \]

and the \(\alpha\)'s are estimated by minimizing

\[ -\ell(\alpha) + \sum_{k} \lambda_k \alpha^T (X^T X)^{-1} \alpha \]

where \(\lambda_k\) are weights.
This is really minimising the functional

$$\sum (y - f)^2 + \lambda \int (s'')^2 dx$$

so for $\lambda \approx 0$, we get an interpolating spline (large df), while for large $\lambda$ (small df), we get a least squares solution.
In the R function `smooth.spline`, we can use `spar (\sim \lambda)` or `df`, both of which control the penalty function, and the position of the knots.

Thus this is an optimisation problem.
Cubic spline, solid = 6df, – – = 12df, ... = 50df

Age in weeks

ME

30 40 50 60 70 80

30 40 50 60 70 80

120 140 160 180 200 220

287
Note the decreasing smoothness with increasing df; the 50df spline is effectively an interpolating spline (dots).
tag207 <- read.table("tag207.dat",header=T)
library(modreg)

spl.mod6 <- smooth.spline(tag207$Age,tag207$ME,df=6)
Kernel smoothers

\[ \hat{y}|x_0 = g(x_0) \]

\[ \hat{y}|x_0 = s(x_0) = \frac{\sum_{i=1}^{n} w_i y_i}{\sum_{i=1}^{n} w_i} \]
The normal type curves shown on the diagram show the weight function used by the kernel. More importance is given to points near the centre, with the weighting dropping off with distance from the centre ($x_0$). The spread of the kernel or importance curve is called the ’bandwidth’. Wide bandwidths oversmooth the data, while very small values interpolate the data. The ideal choice is in between.
Kernel smoothing, solid = bw10, dash = bw5
The wider bandwidth (10) corresponds to the smoother interpolant.
Nonlinear regression
Weight loss over time:
The nonlinear model is:

\[ y = \beta_0 + \beta_1 2^{-t/\theta} + \epsilon, \quad \epsilon \sim N(0, \sigma^2) \]

where

\[ \beta_0 \quad \text{ultimate lean weight or asymptote} \]
\[ \beta_1 \quad \text{total amount to be lost} \]
\[ \theta \quad \text{half life or time taken to lose} \]
\[ \text{half the amount remaining to be lost} \]
library(MASS)
data(wtloss)

library(nls)

nl.mod1 <- nls(Weight ~ b0 + b1*2^(-Days/th), data=wtloss, 
               start=c(b0=90, b1=95, th=120))

summ1 <- summary(nl.mod1)
print(names(summ1))
print(summ1$parameters)

| Estimate | Std. Error | t value | Pr(>|t|) |
|----------|------------|---------|----------|
| b0       | 81.4       | 2.27    | 35.9     | 7.88e-37 |
| b1       | 102.7      | 2.08    | 49.3     | 2.02e-43 |
| th       | 141.9      | 5.29    | 26.8     | 6.24e-31 |

\[ \hat{y}|t = 81.4 + 102.7 \times 2^{-\frac{t}{141.9}} \]
The thin blue line is the kernel estimate and the solid line is the nonlinear regression model.
Grouped data

This is the situation where we want to fit a flexible regression model to data collected over groups, and then test for any group effects.

Example

A computer manufacturer hired a market research firm to investigate the relationship between the chance that a family will purchase a home computer and the price of the home computer. The data that follow are based on replicate surveys done in two cities. One thousand heads of households in each city were randomly selected and asked if they would be likely to purchase a home computer at a given price. Ten prices ($000) were studied, and 100 heads of households in each city were randomly assigned to a given price. The response variable is the proportion likely to purchase at a given price.
A = circle, B = triangle
R Code:

hcdf <- read.table("hcomputers.txt",header=T)
str(hcdf)
attach(hcdf)
plot(ppn ~ price,pch=as.numeric(City),
     main="A = circle, B = triangle")
library(lattice)
xyplot(ppn~price|City)
library(nls)
nl.hc <- nls( ppn~ b0 + b2 * exp(-b3 * price),
          data = hcdf, start=c(b0=0.05, b2=0.7, b3=0.1))
summary(nl.hc)
yf <- fitted(nl.hc)
yf
postscript(file="AByf.ps",horizontal=F,width=6,height=5)
plot(price,ppn)
nl.hc1 <- nls(ppn ~ b0 + b1 * city + b2 * exp(-b3 * price),
  data = hcdf, start = c(b0=0.05, b1=0.01, b2=0.7, b3=0.1))
summary(nl.hc1)
anova(nl.hc, nl.hc1)
yf1 <- fitted(nl.hc1)
yf1
The first model fits a curve ignoring differences between cities:

\[ \hat{y} = b0 + b2e^{-b3price} \]

The data and fitted curve (model 1):
Model 2 allows for a differential shift between cities, but with a common nonlinear response to price, viz

\[ \hat{y} = b_0 + b_1 \text{city} + b_2 e^{-b_3 \text{price}} \]

where \( \text{city} = 0 \) for A, and \( \text{city} = 1 \) for B.

Note the use of the \texttt{anova} function which for \texttt{nls} produces an AOV for differences between models fitted to the same data.

R Output:

```r
> hcdf <- read.table("hcomputers.txt",header=T)
> str(hcdf)
'data.frame': 20 obs. of 4 variables:
$ City : Factor w/ 2 levels "A","B": 1 2 1 2 1 2 1 2 1 2 ...
$ price: num 1 1 2.5 2.5 5 5 10 10 20 20 ...
$ ppn : num 0.9 0.93 0.8 0.77 0.65 0.63 0.46 0.5 0.34 0.3 ...
$ city : int 0 1 0 1 0 1 0 1 0 1 ...
> attach(hcdf)
> plot(ppn ~ price,pch=as.numeric(City),
```
main="A = circle, B = triangle"

> library(lattice)
> xyplot(ppn~price|City)
> library(nls)
> nl.hc <- nls(ppn~b0 + b2 * exp(-b3 * price), data=hcdf,
+       start=c(b0=0.05, b2=0.7, b3=0.1))
> summary(nl.hc)

Formula: ppn ~ b0 + b2 * exp(-b3 * price)

Parameters:

| Parameter | Estimate | Std. Error | t value | Pr(>|t|)  |
|-----------|----------|------------|---------|-----------|
| b0        | 0.083601 | 0.018362   | 4.553   | 0.000282  *** |
| b2        | 0.830532 | 0.026031   | 31.906  | < 2e-16   *** |
| b3        | 0.064120 | 0.006024   | 10.644  | 6.15e-09  *** |

Residual standard error: 0.04054 on 17 degrees of freedom

Correlation of Parameter Estimates:

<table>
<thead>
<tr>
<th>b0</th>
<th>b2</th>
</tr>
</thead>
<tbody>
<tr>
<td>b2</td>
<td>-0.4350</td>
</tr>
</tbody>
</table>
b3  0.7067  0.09824

> yf <- fitted(nl.hc)
> yf
[1] 0.86255099 0.86255099 0.79112209 0.79112209 0.68633046 0.68633046
[7] 0.52101075 0.52101075 0.31396811 0.31396811 0.20492657 0.20492657
[13] 0.14749850 0.14749850 0.11725331 0.11725331 0.09037484 0.09037484
[19] 0.08496437 0.08496437
attr(,"label")
[1] "Fitted values"
> plot(price,ppn)
> lines(price,yf)
> nl.hc1 <- nls( ppn~ b0 + b1 * city + b2 * exp(-b3 * price),
                data = hcdf,
                +            start=c(b0=0.05, b1= 0.01,  b2=0.7, b3=0.1))
> summary(nl.hc1)

Formula: ppn ~ b0 + b1 * city + b2 * exp(-b3 * price)

Parameters:
 Estimate Std. Error t value Pr(>|t|)
 b0 0.0583  0.0157   3.71 0.0003
 b1  0.0099  0.0050   1.98 0.0521
 b2 0.7067  0.0982   7.20 1.8e-12
 b3 0.7067  0.0982   7.20 1.8e-12
b0  0.084601  0.021100  4.010  0.00101  **

b1  -0.002000  0.018680  -0.107  0.91607

b2  0.830532  0.026822  30.964  1.04e-15  ***

b3  0.064120  0.006207  10.330  1.74e-08  ***

Residual standard error: 0.04177 on 16 degrees of freedom

Correlation of Parameter Estimates:

       b0   b1   b2
b0  1.00  -0.4427
b1  -0.4427  1.00  -0.3901
b2  -0.3901  -0.3901  1.00  1.845e-07
b3  0.6337  -2.229e-08  0.09824  1.00

> anova(nl.hc,nl.hc1)
Analysis of Variance Table

Model 1: ppn ~ b0 + b2 * exp(-b3 * price)
Model 2: ppn ~ b0 + b1 * city + b2 * exp(-b3 * price)

          Res.Df Res.Sum Sq Df  Sum Sq F value Pr(>F)
1          17   0.027937

309
> yf1 <- fitted(nl hc1)
> yf1
[1] 0.86355098 0.86155099 0.79212209 0.79012210 0.68733045 0.68533046
[7] 0.52201074 0.52001075 0.31496810 0.31296811 0.20592656 0.20392657
[13] 0.14849850 0.14649851 0.11825330 0.11625331 0.09137484 0.08937485
[19] 0.08596437 0.08396438
attr(,"label")
[1] "Fitted values"

As expected there is no differential shift between cities (P=0.916), as seen in the data plot.
How was the AOV table calculated?

Model 1:

\[
RSE = 0.04054 = \hat{\sigma} \rightarrow \hat{\sigma}^2 = 0.001643 \text{ and so } \hat{\sigma}^2 \times 17 = 0.027939,\]

as per AOV table.

Model 2:

\[
RSE = 0.04177 = \hat{\sigma} \rightarrow \hat{\sigma}^2 = 0.001744 \text{ and so } \hat{\sigma}^2 \times 16 = 0.0279157,\]

as per AOV table.
So the F–test tests the difference between the two models, ie, the city effect via

\[ F = \frac{0.00002}{0.027917/16} = \frac{0.00002}{0.001744} = 0.011467 \]

as per the \texttt{nls} output. Again there is no differential shift between cities.