Stat 8311, Fall 2006, Ordered categories

This is a simple example (one way model with $p = 4, m = 2$) to illustrate fitting with ordered categories. This simply requires replacing the usual basis given by $A \otimes J_2$ with $A = (J_4, e_2, e_3, e_4)$ by different full-rank matrix for A. Let's suppose that the only factor has levels given by the values $1, 2, 3, 5$, so the levels are *not* equally spaced. We first fit the usual way:

```
> set.seed(112)
> y <- rnorm(8)
> x <- c(1, 2, 3, 5)
> (xx <- as.vector(kronecker(x, c(1, 1))))

[1] 1 1 2 2 3 3 5 5

> (m0 <- lm(y ~ factor(xx)))

Call:
lm(formula = y ~ factor(xx))

Coefficients:
(Intercept)  factor(xx)2  factor(xx)3  factor(xx)5
     1.0446      -2.2840      -1.9670      -0.1635

> model.matrix(m0)[, ]

  (Intercept) factor(xx)2 factor(xx)3 factor(xx)5
1           1           0           0           0
2           1           0           0           0
3           1           1           0           0
4           1           1           0           0
5           1           0           1           0
6           1           0           1           0
7           1           0           0           1
8           1           0           0           1
```

This is the usual R parameterization, with $X = A \otimes J_2$.

Suppose we define P by

$$P = \begin{pmatrix} 1 & x_1 & x_1 & x_1 \\ 1 & x_2 & x_2^2 & x_2^3 \\ 1 & x_3 & x_3^2 & x_3^3 \\ 1 & x_4 & x_4^2 & x_4^3 \end{pmatrix}$$

In this particular example, P is given by

```
> model.matrix(~x + I(x^2) + I(x^3))[, ]

  (Intercept) x I(x^2) I(x^3)
1           1 1      1      1
2           1 2      4      8
3           1 3      9     27
4           1 5     25    125
```

We need to use the `I` function to force R to interpret the `^` character correctly. The following model has $X = P \otimes J_2$:

```
> (m1 <- lm(y ~ xx + I(xx^2) + I(xx^3)))

Call:
lm(formula = y ~ xx + I(xx^2) + I(xx^3))

Coefficients:
(Intercept)           xx      I(xx^2)      I(xx^3)
     7.5880      -9.2260       2.9589      -0.2764
```

R computes estimates using the QR decomposition of X. You can do this explicitly with the function `poly`

```
> poly(xx, 3)[, ]

               1          2          3
[1,] -0.41833001  0.3988620 -0.20225996
[2,] -0.41833001  0.3988620 -0.20225996
[3,] -0.17928429 -0.2279212  0.53935989
[4,] -0.17928429 -0.2279212  0.53935989
[5,]  0.05976143 -0.4558423 -0.40451992
[6,]  0.05976143 -0.4558423 -0.40451992
[7,]  0.53785287  0.2849014  0.06741999
[8,]  0.53785287  0.2849014  0.06741999

> (m2 <- lm(y ~ poly(xx, 3)))

Call:
lm(formula = y ~ poly(xx, 3))

Coefficients:
 (Intercept)  poly(xx, 3)1  poly(xx, 3)2  poly(xx, 3)3
    -0.05905       0.40799       2.74125      -0.89449
```

The parameters are essentially uninterpretable when the spacing is not equal. However, getting fitted values is easy, even if the parameters are unclear:

```
> predict(m2, newdata = list(xx = c(0, 1, 2, 3, 4, 5,
+     6, 7, 8, 9, 10)))

          1           2           3           4           5
  7.5880387   1.0445717  -1.2394375  -0.9224109   0.3372294
          6           7           8           9          10
  0.8810612  -0.9493375  -6.8123887 -18.3665147 -37.2701374
         11
-65.1816790
```

This substitutes the new value of `xx` into `poly(xx,3)` to get fitted values.

R has something called an *ordered factor*, but *it assumes that the levels of the factor are equally spaced regardless of the names for the levels of the factor.*

```
> xf <- factor(xx, ordered = TRUE)
> (m3 <- lm(y ~ xf))

Call:
lm(formula = y ~ xf)

Coefficients:
(Intercept)         xf.L         xf.Q         xf.C
   -0.05905     -0.03880      2.04374     -0.24923

> model.matrix(m3)[, ]

  (Intercept)       xf.L xf.Q       xf.C
1           1 -0.6708204  0.5 -0.2236068
2           1 -0.6708204  0.5 -0.2236068
3           1 -0.2236068 -0.5  0.6708204
4           1 -0.2236068 -0.5  0.6708204
5           1  0.2236068 -0.5 -0.6708204
6           1  0.2236068 -0.5 -0.6708204
7           1  0.6708204  0.5  0.2236068
8           1  0.6708204  0.5  0.2236068
```

Using the ordered factor gives a correct basis for E, and hence the overall test will be OK, but the columns of X do not correpond to vectors in E_0 or $E - E_0$, unless spacing is equal.

To test the hypothesis of a linear trend, or a linear and quadratic trend, try any of the following:

```
> anova(m1)

Analysis of Variance Table

Response: y
          Df Sum Sq Mean Sq F value  Pr(>F)
xx         1 0.1665  0.1665  0.1421 0.72538
I(xx^2)    1 7.5144  7.5144  6.4135 0.06449
I(xx^3)    1 0.8001  0.8001  0.6829 0.45504
Residuals  4 4.6866  1.1717

> anova(update(m2, ~xx + poly(xx, 3)))

Analysis of Variance Table

Response: y
            Df Sum Sq Mean Sq F value Pr(>F)
xx           1 0.1665  0.1665  0.1421 0.7254
poly(xx, 3)  2 8.3145  4.1573  3.5482 0.1299
Residuals    4 4.6866  1.1717
```

```
> anova(update(m2, ~xx + poly(xx, 2) + poly(xx, 3)))

Analysis of Variance Table

Response: y
            Df Sum Sq Mean Sq F value  Pr(>F)
xx           1 0.1665  0.1665  0.1421 0.72538
poly(xx, 2)  1 7.5144  7.5144  6.4135 0.06449
poly(xx, 3)  1 0.8001  0.8001  0.6829 0.45504
Residuals    4 4.6866  1.1717
```