General Linear Statistical Models - Part II

Statistics 135

Autumn 2005

What are Factors?

```
> type.lm <- lm(HighFuel ~ Type, data=cars93)
> summary(type.lm)

Residuals:
     Min       1Q   Median       3Q      Max
-0.87891 -0.19098  0.04712  0.22671  0.77217

Coefficients:
            Estimate Std. Error t value Pr(>|t|)
(Intercept)  3.37677    0.08886  38.002  < 2e-16 ***
TypeLarge    0.37248    0.13921   2.676  0.00891 **
TypeMidsize  0.39651    0.11678   3.395  0.00103 **
TypeSmall   -0.49786    0.11795  -4.221 5.95e-05 ***
TypeSporty   0.14754    0.13007   1.134  0.25980
TypeVan      1.20983    0.14809   8.169 2.24e-12 ***
---
```

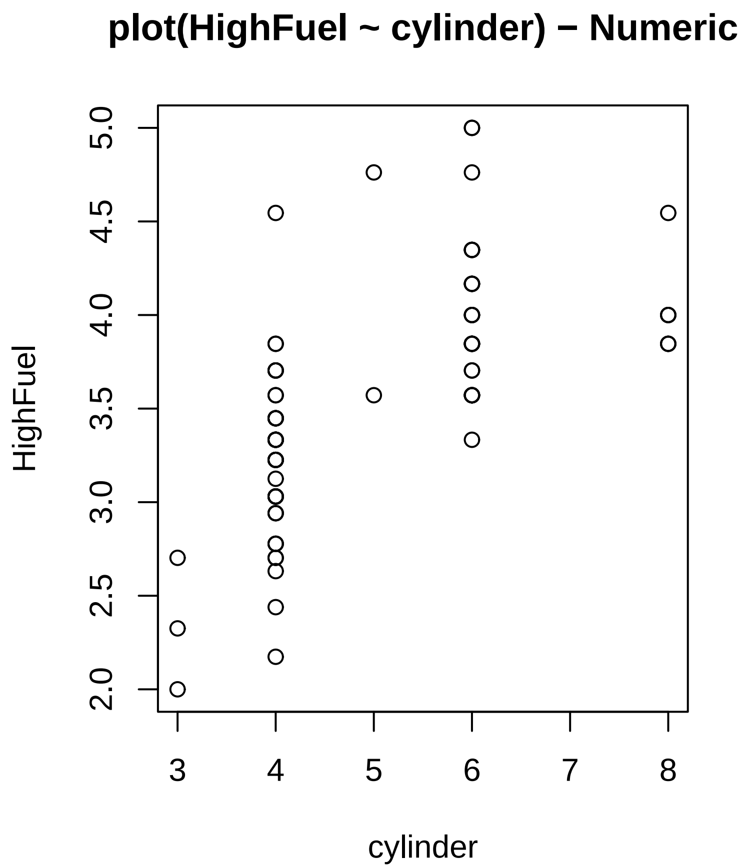
```
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

Residual standard error: 0.3554 on 87 degrees of freedom
Multiple R-Squared: 0.658,      Adjusted R-squared: 0.6383
F-statistic: 33.48 on 5 and 87 DF,  p-value: < 2.2e-16

> anova(type.lm)
Analysis of Variance Table

Response: HighFuel
          Df  Sum Sq Mean Sq F value    Pr(>F)
Type       5 21.1446  4.2289  33.476 < 2.2e-16 ***
Residuals 87 10.9906  0.1263
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

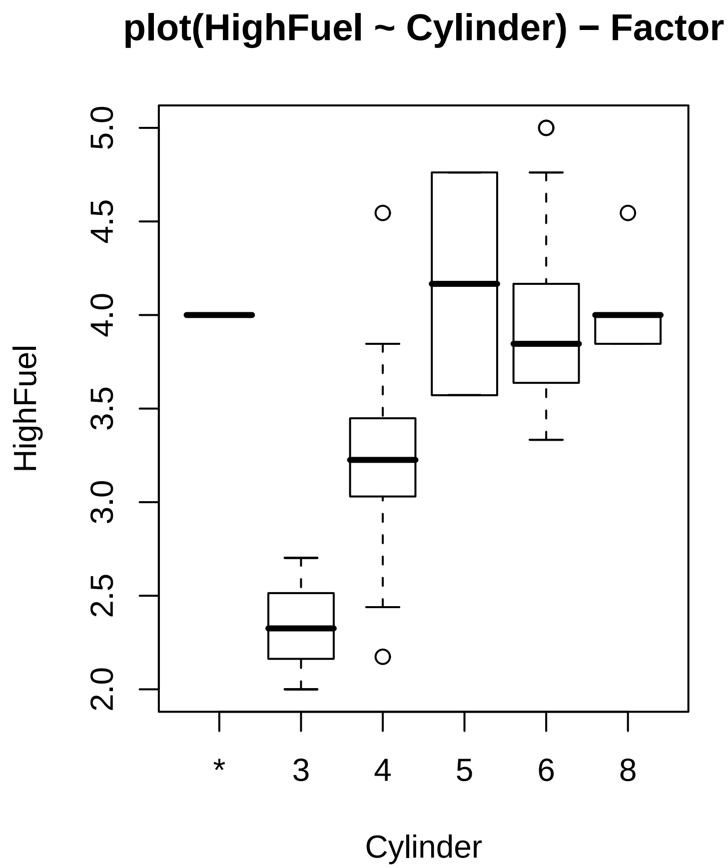
So **R** recognized that `Type` was a factor and created the necessary predictor variables. In this case, it included the indicators for `Large`, `Midsize`, `Small`, `Sporty`, and `Van`. It dropped the indicator variable for `Compact`.

What is a factor?

It is the internal representation of a categorical variable. Character variables, such as `Type` are automatically treated this way. However, numeric variables could either be quantitative or factor levels (or quantitative but you want to treat them factor levels). An example is `Cylinder` which is a factor (in my representation of the data frame). I've created a section version of the variable `cylinder`, which is numeric.

How **S** treats a variable depends of the type.


plot(HighFuel ~ cylinder) − Numeric
HighFuel
2.0
2.5
3.0
3.5
4.0
4.5
5.0
3
4
5
6
7
8
cylinder


plot(HighFuel ~ Cylinder) − Factor
HighFuel
2.0
2.5
3.0
3.5
4.0
4.5
5.0
*
3
4
5
6
8
Cylinder

```
> cylinder.lm <- lm(HighFuel ~ cylinder, data=cars93)
> Cylinder.lm <- lm(HighFuel ~ Cylinder, data=cars93)
> summary(cylinder.lm)

Call:
lm(formula = HighFuel ~ cylinder, data = cars93)

Residuals:
      Min        1Q    Median        3Q       Max
-1.074287 -0.272838  0.001887  0.200075  1.297254

Coefficients:
            Estimate Std. Error t value Pr(>|t|)
(Intercept)   2.0561     0.1853  11.095  < 2e-16 ***
cylinder      0.2980     0.0361   8.257 1.20e-12 ***

Residual standard error: 0.4492 on 90 degrees of freedom
Multiple R-Squared: 0.431,      Adjusted R-squared: 0.4247
F-statistic: 68.17 on 1 and 90 DF,  p-value: 1.202e-12
```

Does a linear regression

```
> summary(Cylinder.lm)
Call:
lm(formula = HighFuel ~ Cylinder, data = cars93)
Residuals:
      Min        1Q    Median        3Q       Max
-1.056216 -0.199826 -0.004322  0.218147  1.315326

Coefficients:
            Estimate Std. Error t value Pr(>|t|)
(Intercept)  4.00000    0.40604   9.851 8.13e-16 ***
Cylinder3   -1.65724    0.46885  -3.535 0.000657 ***
Cylinder4   -0.76987    0.41016  -1.877 0.063870 .
Cylinder5    0.16667    0.49729   0.335 0.738321
Cylinder6   -0.01169    0.41254  -0.028 0.977454
Cylinder8    0.01199    0.43407   0.028 0.978031

Residual standard error: 0.406 on 87 degrees of freedom
Multiple R-Squared: 0.5537,     Adjusted R-squared: 0.528
F-statistic: 21.58 on 5 and 87 DF,  p-value: 5.495e-14
```

Does an ANOVA

The internal representation of a factor is by a numeric vector taking values from 1 to the number of levels. To convert a numeric vector to a factor, you can use `as.factor` function, such as

```
> CylFact <- as.factor(cars93$cylinder)
> CylFact
 [1] 4    6    6    6    4    4    6    6    6    8    8    4    4    6
[15] 4    6    6    8    8    6    4    6    4    4    4    6    4    6
[29] 4    6    4    4    4    4    4    6    6    8    3    4    4    4
[43] 4    4    4    4    4    8    6    6    6    8    4    4    4    6
[57] <NA> 4    6    4    6    4    6    4    4    6    6    4    4    6
[71] 6    4    4    4    6    6    6    4    4    3    4    4    3    4
[85] 4    4    4    4    5    4    6    4    5
Levels: 3 4 5 6 8
```

While the internal coding is from 1 to the number of levels of the factor, they can have other names. To see what the internal coding looks like, use the `as.numeric` function.

```
> as.numeric(CylFact)
 [1]  2  4  4  4  2  2  4  4  4  5  5  2  2  4  2  4  4  5  5  4  2  4  2
[24]  2  2  4  2  4  2  4  2  2  2  2  2  4  4  5  1  2  2  2  2  2  2  2
[47]  2  5  4  4  4  5  2  2  2  4 NA  2  4  2  4  2  4  2  2  4  4  2  2
[70]  4  4  2  2  2  4  4  4  2  2  1  2  2  1  2  2  2  2  2  3  2  4  2
[93]  3
```

The levels can be renamed with the `levels` function. For example, suppose that a vector `religion` took the values 1 for Christian, 2 for Islam, 3 for Judism, 4 for Shinto, and 5 for Flying Spaghetti Monsterism. Instead of showing 1, 2, etc, we can show text labels instead by

```
> religion
 [1] 4 1 4 3 4 1 2 3 4 1 2 2 1 1 3 5 4 1 1 1
Levels: 1 2 3 4 5

> levels(religion) <- c("Christian", "Islam", "Judism", "Shinto", "FSM")
> religion
 [1] Shinto    Christian Shinto    Judism    Shinto    Christian Islam
 [8] Judism    Shinto    Christian Islam     Islam     Christian Christian
[15] Judism    FSM       Shinto    Christian Christian Christian
Levels: Christian Islam Judism Shinto FSM
> levels(religion)
[1] "Christian" "Islam"     "Judism"    "Shinto"    "FSM"
```

How Does `lm` Treat Factors

Lets see what the how the model works for the `Type` example

- `Compact`

$$E[Y|\texttt{Compact}] = \beta_0 + \beta_1 \times 0 + \ldots + \beta_5 \times 0 = \beta_0$$

- `Large`

$$E[Y|\texttt{Large}] = \beta_0 + \beta_1 \times 1 + \beta_2 \times 0 + \ldots + \beta_5 \times 0 = \beta_0 + \beta_1$$

- `Van`

$$E[Y|\texttt{Van}] = \beta_0 + \beta_1 \times 0 + \ldots + \beta_4 \times 0 + \beta_5 \times 1 = \beta_0 + \beta_5$$

So

- $\beta_0 = E[Y|\texttt{Compact}]$

- β_1 is $E[Y|\texttt{Large}] - E[Y|\texttt{Compact}]$

- β_5 is $E[Y|\texttt{Van}] - E[Y|\texttt{Compact}]$

So the parameters $\beta_1, \ldots, \beta_5$ are contrasts of the `Type` means μ_i.

Contrasts for Factors

As mentioned last class, there are different ways of creating predictor variables for categorical factors for the model

$$y_{ji} = \mu + \alpha_j + \epsilon_{ji}$$

Remember that for a factor with k levels, we need $k-1$ variables. **S** has a number of built in ways of handling that. There are 4 different types of contrasts built-in **S**. They are

- `contr.treatment`: Creates indicator variables for each level, except for the first one. This allows for comparing a comparison of each level of the factor with the first. Note that these are **not** actually contrasts. This sets $\alpha_1 = 0$, $\alpha_{j+1} = \beta_j; j < k$, and $\mu = \beta_0$.

```
> options(contrasts=c("contr.treatment","contr.poly"))
> contrasts(cars93$Type)
        Large Midsize Small Sporty Van
Compact     0       0     0      0   0
Large       1       0     0      0   0
Midsize     0       1     0      0   0
Small       0       0     1      0   0
Sporty      0       0     0      1   0
Van         0       0     0      0   1
```

- `contr.sum`: In this parameterization, $\sum \alpha_j = 0$ is enforced with $\alpha_j = \beta_j; j < k$ and $\alpha_k = -\sum \beta_j$. This is a common parameterization in many Design of Experiments / ANOVA texts.

```
> options(contrasts=c("contr.sum","contr.poly"))
> contrasts(cars93$Type)
        [,1] [,2] [,3] [,4] [,5]
Compact    1    0    0    0    0
Large      0    1    0    0    0
Midsize    0    0    1    0    0
Small      0    0    0    1    0
Sporty     0    0    0    0    1
Van       -1   -1   -1   -1   -1
```

- `contr.helmert`: In this parameterization, contrasts of the form $\alpha_1 - \alpha_2$, $\alpha_1 + \alpha_2 - 2\alpha_3$, $\alpha_1 + \alpha_2 + \alpha_3 - 3\alpha_4$. One way of thinking of this is that the first contrast compares the two groups, the second compares the average of the first two with the third, the third compares the average of the first three with the fourth, etc.

```
> options(contrasts=c("contr.helmert","contr.poly"))
> contrasts(cars93$Type)
        [,1] [,2] [,3] [,4] [,5]
Compact   -1   -1   -1   -1   -1
Large      1   -1   -1   -1   -1
Midsize    0    2   -1   -1   -1
Small      0    0    3   -1   -1
Sporty     0    0    0    4   -1
Van        0    0    0    0    5
```

- `contr.poly`: This is used with ordered factors. In some cases, categorical variables have a natural ordering, such as the number of cylinders in a car engine. Most don't. However you might have fun arguing with people on how to order Christianity, Islam, Judism, Shinto, or the Flying Spaghetti Monsterism.

  In the case where order makes sense, **S** has a set of contrasts which allow looking for trends. They are based on orthogonal polynomials, assuming the levels are equally spaced.

```
> cars93$CylinderO <- as.ordered(cars93$Cylinder)
> contrasts(cars93$Cylinder)
  3 4 5 6 8
* 0 0 0 0 0
3 1 0 0 0 0
4 0 1 0 0 0
5 0 0 1 0 0
6 0 0 0 1 0
8 0 0 0 0 1
```

```
> contrasts(cars93$CylinderO)
          .L         .Q         .C         ^4          ^5
* -0.5976143  0.5455447 -0.3726780  0.1889822 -0.06299408
3 -0.3585686 -0.1091089  0.5217492 -0.5669467  0.31497039
4 -0.1195229 -0.4364358  0.2981424  0.3779645 -0.62994079
5  0.1195229 -0.4364358 -0.2981424  0.3779645  0.62994079
6  0.3585686 -0.1091089 -0.5217492 -0.5669467 -0.31497039
8  0.5976143  0.5455447  0.3726780  0.1889822  0.06299408
```

The easiest way for setting the contrasts is with one of the following `options` commands

- `options(contrasts=c("contr.treatment","contr.poly"))` (**R** default)
- `options(contrasts=c("contr.treatment","contr.poly"))`
- `options(contrasts=c("contr.treatment","contr.poly"))` (**S-Plus** default)

By changing the contrast choice, you will get different parameter values, but the **same** fitted values, residuals, R^2, etc. They are all describing the same model, just written out differently.

If you wish to see the current setting, give the command `options("contrasts")`.

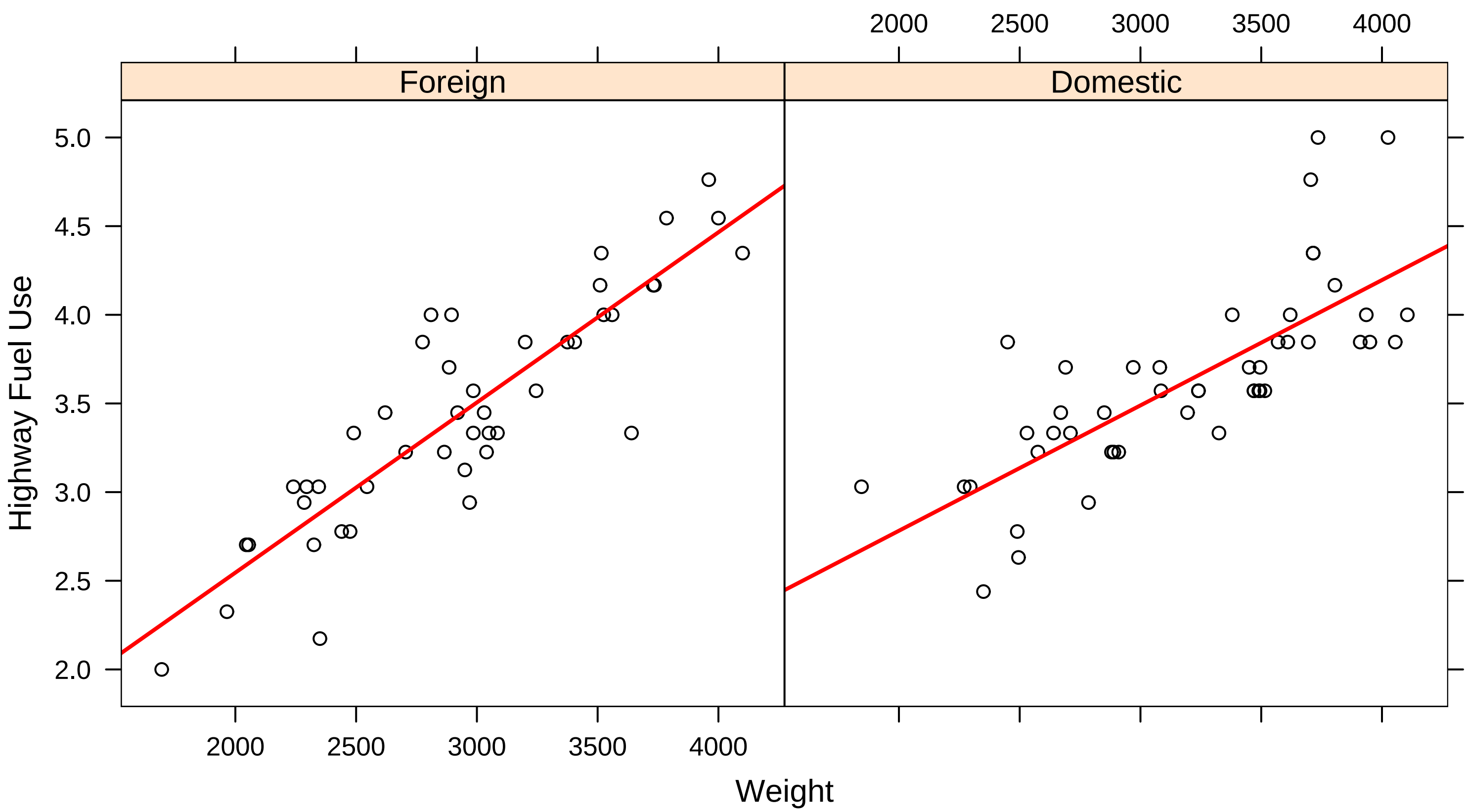
```
> options(contrasts=c("contr.sum","contr.poly"))
> type.sum.lm <- lm(HighFuel ~ Type, data=cars93)
>
> summary(type.lm)

Residuals:
     Min       1Q   Median       3Q      Max
-0.87891 -0.19098  0.04712  0.22671  0.77217

> summary(type.sum.lm)

Call:
lm(formula = HighFuel ~ Type, data = cars93)

Residuals:
     Min       1Q   Median       3Q      Max
-0.87891 -0.19098  0.04712  0.22671  0.77217
```

```
> summary(type.lm)

Coefficients:
            Estimate Std. Error t value Pr(>|t|)
(Intercept)  3.37677    0.08886  38.002  < 2e-16 ***
TypeLarge    0.37248    0.13921   2.676  0.00891 **
TypeMidsize  0.39651    0.11678   3.395  0.00103 **
TypeSmall   -0.49786    0.11795  -4.221 5.95e-05 ***
TypeSporty   0.14754    0.13007   1.134  0.25980
TypeVan      1.20983    0.14809   8.169 2.24e-12 ***

> summary(type.sum.lm)

Coefficients:
            Estimate Std. Error t value Pr(>|t|)
(Intercept)  3.64819    0.03880  94.024  < 2e-16 ***
Type1       -0.27141    0.08228  -3.299  0.00141 **
Type2        0.10106    0.09572   1.056  0.29397
Type3        0.12510    0.07303   1.713  0.09029 .
Type4       -0.76928    0.07427 -10.358  < 2e-16 ***
Type5       -0.12388    0.08672  -1.428  0.15676
```

```
> anova(type.lm)
Analysis of Variance Table

Response: HighFuel
          Df  Sum Sq Mean Sq F value    Pr(>F)
Type       5 21.1446  4.2289  33.476 < 2.2e-16 ***
Residuals 87 10.9906  0.1263
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

> anova(type.sum.lm)
Analysis of Variance Table

Response: HighFuel
          Df  Sum Sq Mean Sq F value    Pr(>F)
Type       5 21.1446  4.2289  33.476 < 2.2e-16 ***
Residuals 87 10.9906  0.1263
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

Interactions

Does the effect of one predictor variable on the response depend of the level of other predictor variables. For example consider



It appears that the relationship between Weight and Fuel Use depends on where the car is made `Domestic`.

If their were no interaction, we would want to fit the additive model

```
HighFuel ~ Weight + Domestic
```

In this case I would at least want to try the interaction model

```
HighFuel ~ Weight + Domestic + Weight:Domestic
```

In **S**, `:` is one way to indicate interactions. There are some shorthands. For example `*` will give the highest order interaction, plus all main effects and lower level interactions. A shorthand for the above is

```
HighFuel ~ Weight*Domestic
```

Suppose we had three variables `A`, `B`, `C`. The model statements

```
y ~ A*B*C
y ~ A + B + C + A:B + A:C + B:C + A:B:C
```

are equivalent. Supppose that you only want up to the second order interactions. This could be done by

```
y ~ (A + B + C)^2
y ~ A + B + C + A:B + A:C + B:C + A:B:C
```

This will omit terms like `A:A` (treats is as `A`)