As mentioned before you need to build your own $X$ matrix for performing regression. For example, to perform the regression in S-Plus/R with

```
lm(BTUIn ~ BTUOut * damper, data=furnace)
```

> summary(furnace.lm2)

Call:
lm(formula = BTUIn ~ BTUOut * Damper, data = furnace)

Residuals:
    Min  1Q Median  3Q     Max
-3.089841 -0.219128  0.003471  0.303370  1.812594

Coefficients:
            Estimate Std. Error t value Pr(>|t|)
(Intercept)   0.16470   0.30425   0.541   0.590
BTUOut       0.92183   0.02755  33.458  < 2e-16 ***
DamperTVD    0.01959   0.43054   0.045   0.964
BTUOut:DamperTVD -0.01717   0.03839  -0.447   0.656

Residual standard error: 0.5576 on 86 degrees of freedom
Multiple R-Squared: 0.9635,  Adjusted R-squared: 0.9622
F-statistic: 756.3 on 3 and 86 DF,  p-value: < 2.2e-16

> anova(furnace.lm2)
Analysis of Variance Table

Response: BTUIn

<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>BTUOut</td>
<td>1</td>
<td>704.66</td>
<td>704.66</td>
<td>2266.766</td>
</tr>
<tr>
<td>Damper</td>
<td>1</td>
<td>0.61</td>
<td>0.61</td>
<td>1.9483</td>
</tr>
<tr>
<td>BTUOut:Damper</td>
<td>1</td>
<td>0.06</td>
<td>0.06</td>
<td>0.2000</td>
</tr>
<tr>
<td>Residuals</td>
<td>86</td>
<td>26.73</td>
<td>0.31</td>
<td></td>
</tr>
</tbody>
</table>
In Matlab you would need to do something like

```matlab
>> xmat = [ones(size(damper)) BTUOut damper BTUOut.*damper];
>> [b, bint, r, rint, stats] = regress(BTUIn, xmat)
>> b

b =

0.1647
0.9218
0.0196
-0.0172

>> bint

bint =

-0.4401  0.7695
 0.8671  0.9766
-0.8363  0.8755
-0.0935  0.0591

>> stats

stats =

0.9635    756.3049    0
```
There is another function that makes doing regression a bit easier in that is will automate some of the setup. The function is `regstat`.

The form of the function is

```
>> regstats(y, data, 'model')
```

`data` is a matrix with each column corresponding to a different variable.

`'model'` can be one of the following strings

- `'linear'` Includes constant and linear terms (default).
- `'interaction'` Includes constant, linear, and cross product terms.
- `'quadratic'` Includes interactions and squared terms.
- `'purequadratic'` Includes constant, linear, and squared terms.
The basic form of the function will bring up a dialog box asking with values you want returned by the function.

One you checked off the variables you want returned, click Calculate Now and a second box will come up asking where you want to store them.
```matlab
>> regstats(BTUIn,[BTUOut damper],'interaction')
>> beta2

beta2 =
0.1647
0.9218
0.0196
-0.0172

>> sqrt(diag(covb2))

ans =
0.3043
0.0276
0.4305
0.0384

Note that the same values are being returned as R.
For programming purposes, there is a second form of the function call that skips the dialog boxes.

>> stattest = regstats(BTUIn,[BTUOut damper], 'interaction',
{ 'beta' 'covb' 'mse'})

To access a component of the output you need to give a command like

>> stattest.covb

ans =
0.0926  -0.0080  -0.0926   0.0080
-0.0080   0.0008   0.0080  -0.0008
-0.0926   0.0080   0.1854  -0.0159
 0.0080  -0.0008  -0.0159   0.0015
```
To help construct dummy variables to include categorical factors into your linear model, there is the `dummyvar` function. It will take a matrix, where the columns correspond to the different factors, and create a set of dummy variables which could tend be used in a regression

```
>> group

group =

    1     1
    1     2
    1     3
    2     1
    2     2
    2     3

>> D = dummyvar(group)

D =

    1     0     1     0     0
    1     0     0     1     0
    1     0     0     0     1
    0     1     1     0     0
    0     1     0     1     0
    0     1     0     0     1
```
Analysis of Variance

There are 3 functions for performing Analysis of Variance in Matlab.

anova1: Balanced 1-way ANOVA
anova2: Balanced 2-way ANOVA
anovan: Unbalanced and higher way ANOVA

In the first 2 functions there must be the same number of observations for each treatment combination. If this doesn’t hold, you must use anovan.

The following example is taken from a study by Hogg and Ledolter (1987) of bacteria counts in shipments of milk.

>> hogg

hogg =

24  14  11   7  19
15   7   9   7  24
21  12   7   4  19
27  17  13   7  15
33  14  12  12  10
23  16  18  18  20

The desired form for the data is for each column to correspond to a different factor levels. To for this example, column 1 is 6 observations from shipment 1, column 2 is shipment 2, etc.
>> [p1, tbl1, stats1] = anova1(hogg)

p1 =
    1.1971e-004 % p-value from F test

tbl1 = % ANOVA table

'Source'   'SS'    'df'  'MS'    'F'     'Prob>F'
'Columns'  [ 803.000]   [4]  [200.750]  [9.0076]  [1.1971e-004]
'Error'    [ 557.167]   [25]  [ 22.287]    []      []
'Total'    [1.3602e+003] [29]  [  ]      [ ]      [ ]

stats1 = % ANOVA study information

  gnames: [5x1 char]
  n: [6 6 6 6 6]
  source: 'anova1'
  means: [23.8333 13.3333 11.6667 9.1667 17.8333]
  df: 25
  s: 4.7209

In addition to the saved output, anova1 generates two figures, one with the ANOVA table and the other with boxplots of the observations.
Given the highly significant $F$ test, it would be nice to figure which shipments are different. This can be done with the multcompare function. This function takes the ANOVA study information from the anova functions and compares the different groups, taking account of the different number of tests involved.
>> [cal, mal] = multcompare(statsa1)

\[cal\] =

\[
\begin{array}{cccccc}
1.0000 & 2.0000 & 2.4953 & 10.5000 & 18.5047 \\
1.0000 & 3.0000 & 4.1619 & 12.1667 & 20.1714 \\
1.0000 & 4.0000 & 6.6619 & 14.6667 & 22.6714 \\
1.0000 & 5.0000 & -2.0047 & 6.0000 & 14.0047 \\
2.0000 & 3.0000 & -6.3381 & 1.6667 & 9.6714 \\
2.0000 & 4.0000 & -3.8381 & 4.1667 & 12.1714 \\
2.0000 & 5.0000 & -12.5047 & -4.5000 & 3.5047 \\
3.0000 & 4.0000 & -5.5047 & 2.5000 & 10.5047 \\
3.0000 & 5.0000 & -14.1714 & -6.1667 & 1.8381 \\
4.0000 & 5.0000 & -16.6714 & -8.6667 & -0.6619
\end{array}
\]

\[mal\] =

\[
\begin{array}{cc}
23.8333 & 1.9273 \\
13.3333 & 1.9273 \\
11.6667 & 1.9273 \\
9.1667 & 1.9273 \\
17.8333 & 1.9273
\end{array}
\]

The first variable output given confidence intervals for the difference in means for each treatment comparison

\[
\begin{array}{ccccc}
2.0000 & 4.0000 & -3.8381 & 4.1667 & 12.1714 \\
\text{Level 1} & \text{Level 2} & \text{Lower CI} & \text{Est} & \text{Upper CI}
\end{array}
\]

The estimate of the difference is the first level minus the second level.

The second variable output gives the estimated treatment effects with their standard errors.
In addition the function gives a graphical summary indicating which groups are significantly different. This summary gives a graphical summary of the CI’s for each treatment. Clicking on a interval will show which treatment are different than the one of interest.

There are many different multiple comparison procedures built into this function. Which one you use depends on the sort of comparisons you are interested.

The default is Tukey’s HSD (honestly significant difference), which is optimal for balanced designed where all pairwise comparisons are being made.
Also available are lsd, bonferoni, dunn–sidak, and scheffe. These may be more appropriate. For example, if there are other treatment contrasts of interest, scheffe may be more appropriate as it provides a simultaneous confidence level for all possible treatment contrasts.

Not surprisingly, you can set the significance level for the multcompare. The default level is 0.05.

anovan:

This can be used to fit more complicated ANOVA designs involving more that 2 factors and with unbalanced data.

The structure of the functions call is

\[ [p, \text{table}, \text{stats}, \text{terms}] = \text{anovan}(y, \{\text{fac1 fac2 … facn}\}, \text{model}, \text{sstype}, \text{‘varlabels’}) \]

- \( y \): response variable
- faci: factor levels
- model: model to be fit
- sstype: sum of squares type (1, 2, 3)
- varlabels: factor variable names
p: p-values form each $F$ test

table: ANOVA table as a cell array

stats: ANOVA summary output (not particularly readable

terms: model fit

The model can be specified in many ways. Possibilities are

‘linear’ main effects only

‘interaction’ main and two way interactions

‘full’ all interactions up to order $n$

A single number indicates the maximum order of interactions to be fit. So 1 is equivalent to ‘linear’, 2 with ‘interaction’, and $n$ with ‘full’. A choice of 3 would fit the model with all main effects, all 2 and 3 factor interactions. It is also possible to fit more complicated models, such as $A + B^*C$ using S notation. However is a bit kludgy as it is based on the base 2 representations of the numbers.

Let 1 represent fac1

Let 2 represent fac2

Let 4 represent fac3

Let $2^k$ represent fac$k$
To include an effect in the model, you need to add the numbers corresponding to the factors in the effect.

For example, for the furnace data set:
- type: 1
- chshape: 2
- chliner: 4

So to fit the model type * chshape * chliner (= type + chshape + chliner + type:chshape) you would need the terms 1, 2, 4 plus (3 = 1 + 2).

The call would be:

```matlab
>> [pn, tablen, statsn, termsn] = anovan(BTUIIn,[type chshape chliner],[1 2 4 3],1, varnames)
```

Note that the model indicated by [1 2 4 3] is not the same as [3 4] as this would not included the main effects for factors 1 and 2.
In addition to the output variables, anovan also returns the ANOVA table as a figure.

### Sequential (Type I) Sums of Squares

This model was fit using Sequential (Type I) sums of squares. The default choice is Type III. For balanced designs, it doesn’t matter which you use. However for unbalanced designs, it does.

<table>
<thead>
<tr>
<th>Source</th>
<th>Sum Sq.</th>
<th>d.f.</th>
<th>Mean Sq.</th>
<th>F</th>
<th>Prob&gt;F</th>
</tr>
</thead>
<tbody>
<tr>
<td>Type</td>
<td>8.351</td>
<td>2</td>
<td>4.1757</td>
<td>0.54</td>
<td>0.5863</td>
</tr>
<tr>
<td>ChimShape</td>
<td>50.271</td>
<td>2</td>
<td>25.1357</td>
<td>3.24</td>
<td>0.0446</td>
</tr>
<tr>
<td>ChinLiner</td>
<td>8.088</td>
<td>2</td>
<td>4.0442</td>
<td>0.52</td>
<td>0.5962</td>
</tr>
<tr>
<td>Type*ChimShape</td>
<td>32.155</td>
<td>4</td>
<td>8.0387</td>
<td>1.03</td>
<td>0.3946</td>
</tr>
<tr>
<td>Error</td>
<td>605.832</td>
<td>78</td>
<td>7.7571</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Total</td>
<td>704.698</td>
<td>88</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

### Contained (Type III) Sums of Squares

In unbalanced designs, the $F$ tests reported in the table, are different for the different sums of squares types.

<table>
<thead>
<tr>
<th>Source</th>
<th>Sum Sq.</th>
<th>d.f.</th>
<th>Mean Sq.</th>
<th>F</th>
<th>Prob&gt;F</th>
</tr>
</thead>
<tbody>
<tr>
<td>Type</td>
<td>14.011</td>
<td>2</td>
<td>7.0055</td>
<td>0.9</td>
<td>0.41</td>
</tr>
<tr>
<td>ChimShape</td>
<td>5.743</td>
<td>2</td>
<td>2.87153</td>
<td>0.37</td>
<td>0.5921</td>
</tr>
<tr>
<td>ChinLiner</td>
<td>5.977</td>
<td>2</td>
<td>2.98836</td>
<td>0.39</td>
<td>0.5819</td>
</tr>
<tr>
<td>Type*ChimShape</td>
<td>32.155</td>
<td>4</td>
<td>8.0387</td>
<td>1.03</td>
<td>0.3946</td>
</tr>
<tr>
<td>Error</td>
<td>605.832</td>
<td>78</td>
<td>7.76707</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Total</td>
<td>704.698</td>
<td>88</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
Analysis of Covariance

The function aoctool is useful examining models involving 1 continuous predictor and 1 categorical predictor. It allows for all possible models involving these two predictors to be examined in an interactive format.

The function starts off by fitting the full interactions model \((y \sim A \times B)\), but you can fit the models \((y \sim 1, y \sim A, y \sim B, y \sim A + B)\) as well.

The form of the function call is

\[
aoctool(y, \text{cont}\_\text{pred}, \text{cat}\_\text{pred})
\]

\[
>> \text{aoctool(BTUOut,BTUIn,damper)}
\]
Figure No. 1: ANOCOVA Prediction Plot

Figure No. 2: ANOCOVA Test Results

Figure No. 3: ANOCOVA Coefficients
When a different model is request by changing the model menu, all figures update automatically.
It is possible to focus on an individual groups and predictions for given level of the continuous predictor.