Common Confusion 13
Regression
Matthew Woodring

Regression is one of the topics covered towards the end of the course and students tend to find it difficult. In this course, we deal with three types of trendline models: linear, power, and exponential. You must treat each of these models differently in order to get a good model, and more importantly, the correct answer.

The first thing you must understand in order to use regression in MATLAB is the 'polyfit' function. This function takes three inputs: x-data, y-data, and the highest order polynomial. MATLAB will attempt to best-fit a polynomial, of the highest order, to your data. In this course, the highest order polynomial is always '1'. For example, if you had some x-data called 'xValues' and some y-data called 'yValues', with a roughly linear relationship, you would call the following function:

'C = polyfit(xValues, yValues, 1)'

This would result in 'C' being a [2x1] vector. The first item in 'C' could be '7' and the second item in 'C' could be '5'. This would be equivalent to the polynomial:

'y = 7x + 5'

If you recall the equation for the slope of a line, y = mx + b, you will recognize that 'm' is equal to '7' and 'b' is equal to '5'. In summary, for this example:

'C = [m, b]'
and
'm = 7 = C(1)
and
'b = 5 = C(2)'

For linear models, the 'polyfit' model is straightforward because a highest order polynomial of '1', which we will always use in this course, corresponds to a linear equation (y = mx + b). However, as we will see, special care must be given to the power and exponential models.

For a power model, we cannot just plug in our x-values and y-values into the 'polyfit' function with a highest order polynomial of '1'. This is because a highest order polynomial of '1' corresponds to a linear model, not a power model. Therefore, we must transform our power

model into a linear model and then use the ‘polyfit’ function with a highest order polynomial of ‘1’. Recall that a power model is in the form of:

$$‘y = bx^m’$$

To transform this, we will simply use algebra and the rules of logarithms:

$$‘y = bx^m’$$
$$‘\log10(y) = \log10(bx^m)’$$
$$‘\log10(y) = \log10(b) + \log10(x^m)’$$
$$‘\log10(y) = \log10(b) + m*\log10(x)’$$
$$\mathbf{‘\log10(y) = m*\log10(x) + \log10(b)’}$$

Now, we have our transformed linear model which can be plugged into the ‘polyfit’ function. We can call the ‘polyfit’ function as:

$$‘C = \text{polyfit}(\log10(\text{xValues}), \log10(\text{yValues}), 1)’$$

This would result in ‘C’ being a [2x1] vector. The first item in ‘C’ could be ‘7’ and the second item in ‘C’ could be ‘5’. This would be equivalent to the polynomial:

$$‘y = 7x + 5’$$

If you recall the equation for the slope of a line, y = mx + b, you will recognize that ‘m’ is equal to ‘7’ and ‘b’ is equal to ‘5’. However, recall that our ‘b’ value has been transformed by an order of ‘log10’. Therefore, when pulling the ‘m’ and ‘b’ values out of ‘C’, we must account for this by canceling out the ‘log10’ by raising the value of ‘b’ to the power of ‘10’. In summary, for this example:

$$‘C = [m, \log10(b)]’$$
and
$$‘m = 7 = C(1)$$
and
$$‘b = 5 = 10^{C(2)}’$$

Common Confusion 13
Regression
Matthew Woodring

For power models, we still use a highest order polynomial of ‘1’ in the ‘polyfit’ function, but we must account for the fact that a power model is not a linear model. To do this, we do as above and transform the power model into a linear model.

For an exponential model, we cannot just plug in our x-values and y-values into the ‘polyfit’ function with a highest order polynomial of ‘1’. This is because a highest order polynomial of ‘1’ corresponds to a linear model, not an exponential model. Therefore, we must transform our exponential model into a linear model and then use the ‘polyfit’ function with a highest order polynomial of ‘1’. Recall that an exponential model is in the form of:

$$‘Y = be^{mx}’$$

To transform this, we will simply use algebra and the rules of logarithms. Recall that the ‘log’ function in MATLAB is equivalent to the natural log (ln):

$$‘Y = be^{mx}‘$$
$$‘\log(y) = \log(be^{mx})’$$
$$‘\log(y) = \log(b) + \log(e^{mx})’$$
$$‘\log(y) = \log(b) + mx’$$
$$\mathbf{‘\log(y) = mx + \log(b)’}$$

Now, we have our transformed linear model which can be plugged into the ‘polyfit’ function. We can call the ‘polyfit’ function as:

```
‘C = polyfit(xValues, log10(yValues), 1)’
```

This would result in ‘C’ being a [2x1] vector. The first item in ‘C’ could be ‘7’ and the second item in ‘C’ could be ‘5’. This would be equivalent to the polynomial:

$$‘y = 7x + 5’$$

If you recall the equation for the slope of a line, y = mx + b, you will recognize that ‘m’ is equal to ‘7’ and ‘b’ is equal to ‘5’. However, recall that our ‘b’ value has been transformed by an order of ‘log’. Therefore, when pulling the ‘m’ and ‘b’ values out of ‘C’, we must account for this by canceling out the ‘log’ by taking the exponential of ‘b’. In summary, for this example:

Common Confusion 13
Regression
Matthew Woodring

‘C = [m, log(b)]’
and
‘m = 7 = C(1)
and
‘b = 5 = exp(C(2))’

For exponential models, we still use a highest order polynomial of ‘1’ in the ‘polyfit’ function, but we must account for the fact that an exponential model is not a linear model. To do this, we do as above and transform the exponential model into a linear model.

Hopefully, after reading this file, you understand the ‘polyfit’ function and the idea of regression better. These topics will likely appear on the final exam and the idea of regression will appear again if you ever take a statistics course. Please refer to the ‘13. PolyfitModelsReference.pdf’ file on my GitHub for a summary of the topics contained in this file.