**Regression**

Now let’s take a look at curve fitting. Makes an interesting point about why arbitrary curve fitting is not objectively valuable. Given a data set, you can precisely fit all the data if have polynomial curve of high enough degree. But question is, if you generate experimental data again, from the same experiment, will that same curve explain *that* data well? And typically, the answer is emphatically no. What will typically explain the data best, is a best fit based on a curve model that does actually reflect the underlying behavior of the data. Problem is that the higher polynomials will end up fitting the ‘noise’ rather than the ‘phenomenon’. So to test whether a model is fitting the phenomenon well, you should ‘cross validate’, which is split your data up into groups, curve fit to one of the data sets, and then compute the R2 (see below) for the other data sets with that same curve.

**Linear Regression**

General idea is to say that we have a sequence of random variables, Yi. We may suppose that these can be put in one-to-one correspondance with a set of non-random variables, xi, via some relationship such as the following,



etc., where εi are presumed to be identically normally distributed variable with zero mean and variance σ2. And m, b are constant (non random) fit parameters, which the Yi are linearly related to. Note that since εi is presumed normally distributed, this would make all the Yi’s normally distributed too. And Yi’s mean and variance would be: <Yi> = mxi + b + εi = mxi + b. And <Yi>2 = <mxi+b>2 + <εi>2 = 0 + 1 = σ2. Note linear regression requires Yi to be linearly related to the *fit parameters*, not to the independent variable xi. So we could say f(x) = m·ln(x) + b for instance, and just rename ln(x) as some w, and proceed as follows. But we’ll stick with the former guy for now, as it’s more common. So given some plot of data,

Chart, scatter chart

Description automatically generated

we’d like to work out what m and b are. Apparently an unbiased estimator for m and b can be found using least squares technique. We form the so-called sum-of-squared-errors random variable,



and find the m and b in f which would minimize its expectation. It makes sense that minimizing the SSEf will find the best fit parameters, just as minimizing the regular variance will locate the mean. Working this out for our simple linear model, we have (leaving off the implicit expectation brackets, < >, as they don’t really matter to the algebra:



and then we’d minimize this w/r to m and b.



and,



define:



and our two equations come to:



And our solution is, from Cramer’s rule:



**Point estimator for m and b**

So we find, promoting our letters to capital letters since they’re really random variables,



Note that the formula for m is essentially the Covariance of the x and y data points, divided by the variance of the x data points. But remember x is not a random variable per se´, and while Yi is a random variable with its own distribution, the set {Yi} doesn’t really have a probability distribution in aggregate. So that’s why we’re writing and instead of <x> and <Y>. FWIW, our equation is:



which shows that the regression line goes through the sample average and sample average , interestingly. Anyway, it’s importants to keep in mind that for any given set of values {yi}, we can form these estimates for m and b. But since Yi is a random variable, the next set of measurements could give us a new set of yi’s. So really, M and B have probability distributions with their own statistics. So really, M and B have probability distributions with their own statistics. In fact, they are Gaussian distributed, since they are linear combinations of Yi variables, which are themselves Guassian random variables. Let’s work out the expectation and variance of these random variables. We will be using Y­I = f(xi) + εi = mxi + b + εi, from which it follows <Yi> = mxi + b, <Yi>2 = 1, etc.



So <M> = m, which means it’s an unbiased predictor of m. As for B, we’d have:



So B is an unbiased predictor as well. What about the variances?



Not feeling like doing this, but we can calculate the variance of B as well. Have to be careful because M contains Yi’s in it. But all total, we end up with:



Can see that the more data points we have, the smaller the variance, and the better assured we can be of the fit. Note that if we assume Y is normally distributed (i.e., that ε is normally distributed), then both M and B are as well, with the given means and variances. To make practical use of these formulas it is nice to have a way to estimate σ2. Turns out an unbiased estimator of σ2 comes from SSEf.



and can show that:



Anyway. So if we know know σ2 for sure, then we can say these random variables,



follow a unit normal distribution. And if we don’t know it for sure (not sure how we would), we could still use these formulas to good accuracy, if n > 30 or so, by estimating σ2 with <SY2>. But if n < 30 ish, then it might be better to form the Z-sample statistic,



which follows a Student’s T distribution with ν = n-2 d.o.f. If ν > 30 or so, then this is just a normal distribution for all intents.

**Hypothesis Testing of Parameters**

If we are running an experiment, and get some particular value, <M> = m, <B> = b for our linear regression, we may wish to see whether our results invalidate someone else’s linear regression values, or the standard/accepted linear regression values, m\* and b\*. We can do a hypothesis test for this. We know how M and B are distributed, according to our experiments, and so I guess we could form the Z-statistics,



and then calculate the p-values, the probabilities that values at least as extreme as m\* and b\* occur. So we’d calculate,



And p(z) would be a Student’s T distribution with ν = n-2 d.o.f. I can’t stand orphaned headings. So typing this. Ahhh. Better.

**Confidence Intervals for Parameters**

We can also calculate confidence intervals for the statistics M and B. Just like we’ve done with other statistics, we’d say, at the 1-α confidence level,



And zα would be for the aforementioned Student’s T distribution.

**Goodness of Fit: R2 value**

R2 is a measure of the goodness of fit, of the regression curve. It’s given by:



where fi are the regression curve data points, which one could say is a random variable since it depends in known way (if have formula) on the random variable Yi, the data points we’re trying to fit. SSEf = sum of squares about f, and SSEm = sum of squares about mean. This is actually a general formula that applies to all curve fits. Apropos the fraction, the numerator is the ‘variation around the line, or f’, and the denominator is the ‘variation around the mean’). Can see R2 = 1 if there is no variation around the line, i.e., if the curve fits the data exactly. On the other hand, if the curve fits the data no better than the mean, then we get R2 = 0. If you have an R2 = 0.75, then you can say that 75% of the variation of the data is explained/predicted by the regression curve f. To make sense of that, consider following typical situation:

Chart, scatter chart

Description automatically generated

The green line represents the residual/variation in y about its mean . Could say that the purple line is the amount of this variation that is ‘explained’ by the regression curve *f*. Then the pink line would be the variation in y that itsn’t explained by *f*. This is just for heuristic purposes. So we’ll say:



Can therefore write R2 as:



Terminology note: Adjusted R2 = R2 – penalty for number of fitting parameters you have.

Last, should point out that it is possible to get R2 < 0, and even really less than 0, like R2 = -5.2 or something. This will happen if the regression formula, f, fits the data worse than the mean . This shouldn’t happen when you apply a regression formula to the data it was trained on, i.e., the data that was used to create the regression formula, but it can happen if you apply that regression formula to a different set of data (like the testing data).

Okay let’s work out a formula for R2 in our univariable linear regression situation. The R2 formula would be:



Guess we can fill in m from above,



So we see that,



Might note this is Cor(x,Y), if we were to treat the data points x and y as random variables. Terminology note: Multiple R2 is just plain old R2.

**Hypothesis Testing Different Regression Models**

Say you find a trend in the data, via all the regression stuff above. But someone else says that the data is actually just randomly distributed about the mean, and there really is no trend. How could we test this hypothesis? Basic way is this. Let’s define a Null Hypothesis.

H0 = assumption that the data is described by model **Y** = μ + ε, i.e., that m = 0.

And note SSEm be the sum of the square errors for this model.And might also note that this is just a linear regression model with m = 0. Then let’s compare to our full linear model with f = 2 degrees of freedom, fitting **Y** = X**v**(f) + ε. And let SSEf be its sum of squared errors. We would anticipate this to be smaller of course, i.e., SSEm > SSEf. The alternative hypothesis would be:

HA = assumption that the m parameter in the new model Y = Xv(f) + ε is non-zero.

Turns out SSEf follows a known probability distribution. Well we can form a test statistic,



and n = number of data points, and again f = 2. We might recall SSEf/(n-2) is just the point estimator for σ2 in the f-model. Might interpret Z as:



Turns out this follows an F-distribution (B is the β function).



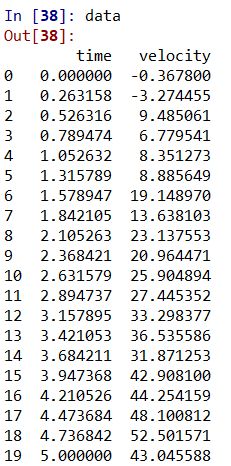
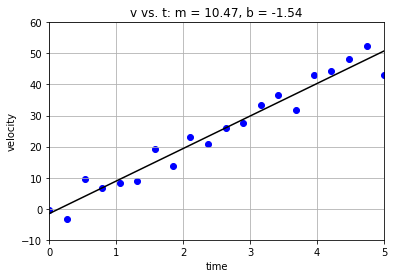
which is the probability density of getting an Z-value of x, given the null hypothesis is true. So we can calculate a p-value,



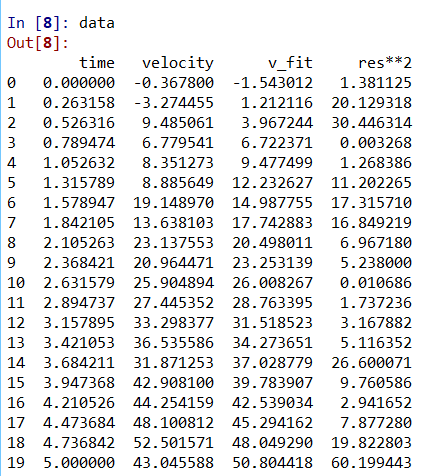
which would be the probability that we’d get an Z-statistic value of Z\* or higher, out of the new model, if the Null hypothesis were true. So if the f model has true explanatory power, then we should find Z\* >> 0 and the p-value should be small (less than 0.05 at the 95% significance level) – we’ll observe that Z grows as SSEf shrinks. I guess we could say that the p-value is the probability we should find as small variation about our best fit line as we do, if the true model is just the mean.

**Example**

Let’s consider doing simple linear regression on the speed of a falling object. We have this data, on the left, which I plotted and linearly regressed, on the right.

According to standard physics, our slope and intercept ought to be 9.8 and 0 respectively. So do our results invalidate the theory? Let’s do a hypothesis test. We’ll calculate a one-tailed test, the probability we’d get m\* < 9.8, given m = 10.47. And probability we’d get b\* > 0, given b = -1.54. M and B are Gaussian distributed, but we don’t know σ2. I’ll estimate it though. So I found the residuals between the data points and the line, and squared them: res\*\*2 = (velocity – v\_fit)2.



Then we have:



and from this, we can find, performing the requisite actions on the x (time) column,



And now we’ll calculate our p-values.



Since our p-values are greater than 0.05, we typically would fail to reject our Null Hypothesis, that m = 10.47 and b = -1.54. By the same token, if we had taken the m\* = 9.8, and b\* = 0 values as the ‘Null’, then our results here would fail to invalidate m\* and b\*. Now in reality, since we have n = 20 data points, we shouldn’t be using a normal distribution to calculate p-values, but a Student’s T distribution. So let’s recalculate, with ν = n – 2 = 18 d.o.f.,



So results aren’t much different, but consistent with the fact that the Student’s T distribution has fatter tails. Now let’s get confidence intervals for our parameters. I’ll do 1 – α = 95% confidence interval, using the Student’s T distribution with ν = n-2 = 18 d.o.f.



So the true values are within our confidence intervals. Just for the sake of doing the last thing on the page, let’s find the p-value for using a linear fit vs. a horizontal mean fit. This will tell us, more or less the odds of getting variation about our line as small as we do if the true model were just the mean. So we calculate,



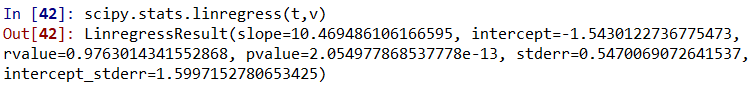
This seems like a lot. And our p-value is:



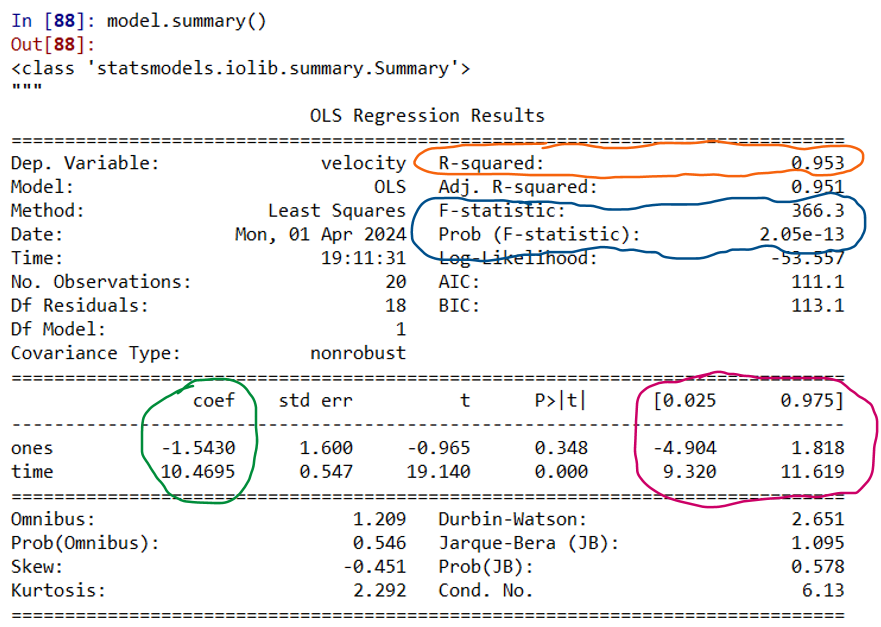
using scipy to do the F-distribution integral. So basically, there is no chance our data is described by just the mean. This quantifies what is pretty clear from the graph. Last we’ll do R2,



which corresponds to an r = √R2 = 0.976. Much of this information is available directly from the linregress function in scipy.



Our p-value is a *little* off. Its stderr is presumably a 68% confidence interval, which should half of our 95% confidence interval, and it is. Same with the intercept stderr. And our r-value matches theirs. So everything checks out. Might be a little better to do it using the *statsmodels* package. Here’s the readout for our linear regression. We’ll recognize the R2 guy, and the F stuff we calculated (the Z-value and p-value). And then we have the linear regression coefficients, as well as the 95% confidence interval bounds.



**Appendix: Outliers**

Let’s consider outliers. I think we can say that outliers are data which are either faulty, in which case they should be removed, or they’re extreme fluctuations, or they’re seemingly outliers because we don’tt have the right independent variables that would show them not to be. In that case, I think they still should be removed, because through the lack of the requisite independent variables, our/any model wouldn’t have the explanatory power to incorporate them. And so we should just focus on what we can explain. So how do we detect outliers? Consider the graph,

A graph of a line with dots and numbers

Description automatically generated

An outlier would be considered something like this blue point, below:

A red line with black dots

Description automatically generated

Note we wouldn’t be able to detect it based on its x or z value per se´. As the z-value is within the range of other z values – approximately in the middle of the range in fact. And the x-value is also within the range. What makes it an outlier is that it is so far off the regression line. Do we have to have a regression line before we can classify something as an outlier? Maybe we could break the data up into bins, Δx, and calculate the standard deviation, σΔx, of points within the interval. Then if there is an interval where σΔx is exceedingly large, perhaps there is an outlier there. That probably works in principle. But in practice, it’s easiest to run an algorithm off of a regression line. What seems to work best is:

1. Draw a regression line through all points, R0.
2. Identify outliers based off of R0. Take these out, and put them in a set S1. Using the quartile identification method, (Q1 – 1.5×IQR, Q3 + 1.5×IQR) seems best.
3. Redraw a regression line, R1, through remaining points.
4. Put outliers, S1, back in, and identify new outliers based off of R1. We’ll call these S2. And take these points S2 out.
5. Redraw a regression line, R2, through remaining points.
6. Put outliers, S2, back in, and identify new outliers based off of R2. We’ll call these S3. And take these points S3 out.
7. Redraw a regression line, R3, through remaining points.
8. Put outliers, S3, back in, and identify new outliers based off of R3. We’ll call these S4, etc.
9. And keep doing this until the outlier sets converge to a final result, i.e., Sn = Sn-1. And we’d say our final regression curve is Rn.

**Appendix: Polynomial Regression**

Say we have dataframe, X,

|  |
| --- |
| x1 |
| 1.2 |
| 3.9 |
| 4.2 |

If wanted to do, say, quadratic regression on X, instead of linear regression, then we could just add a column,

|  |  |
| --- | --- |
| x1 | x12 |
| 1.2 | 2.4 |
| 3.9 | 15.2 |
| 4.2 | 17.6 |

and do linear regression on *this*.