Understanding and Applying Linear Regression

by Vitthal Srinivasan

Start Course

Bookmark Add to Channel Download Course

Course Overview

Course Overview

Hi everyone. My name is Vitthal Srinisvasan. Welcome to my course, Understanding and Applying Linear Regression. I am co-founder of the start-up named Loonycorn, and before this I worked at Google and studied at Stanford. Linear regression is really the first crossover hit from machine learning. It became a mainstream technique taught widely in business schools and in courses on psychology, finance, and many other disciplines far before machine learning had achieved the popularity that it has today. See how linear regression is widely used in forecasting and quantifying cause/effect relationships. Some of the major topics that we will cover include building regression models in Excel, Python, and R, explaining variance and forecasting using regression, the use of categorical variables, and the interpretation of a regression statistic such as the R-squared, standard errors, E-statistics, and F-statistics. By the end of this course, you'll know how to build the robust well-designed regression models in any one of three powerful tools, Excel, R, and Python. You will apply this knowledge to explaining the performance of financial stocks and see how flaws in regression models can be identified and fixed. You will also know how to incorporate categorical variables into these models the right way. Before beginning the course, you should be familiar with at least one out of Excel, Python, and R. From here, you should feel comfortable diving into courses on topics such as factor analysis, principal component

analysis, logistic regression, and machine learning. I hope you'll join me on this journey to learn how to connect the dots with the course, Understanding and Applying Linear Regression at Pluralsight.

Modeling Relationships Between Variables Using Regression

Connecting the Dots Using Linear Regression

Hello, and welcome to this course on Understanding and Applying Linear Regression. We are going to begin this course with a look at using regression models as a way to connect the dots and I mean that literally. We will see how to set up data in the form of a regression problem, that's the essential prerequisite for using regression, and then go on to understand why regression is such a popular tool. We will also understand how regression can be seen as one of the earliest and most popular forms of machine learning. So let's start at the beginning. Let's start with a discussion of how regression is a tool for connecting the dots. My mind is made up. Don't confuse me with facts. Unfortunately, all too often, this is the attitude that decision makers in many organizations have and data professionals have got to fight and overcome this attitude. Indeed, an essential attribute of a successful data professional these days is the ability to deliver thoughtful, fact-based points of view. And so, as a data professional, really your job has only just begun when you've painstakingly collected a ton of data. You've then got to think about it and come up with a thoughtful perspective which balances pros and cons and that perspective needs to be sharpened into a point of view, a crisp call to action. And this really explains the power of regression because regression helps a data professional to connect the dots. The two most common applications of regression are in explaining variance and in making predictions. Explaining variance is all about quantifying cause-and-effect relationships. Making predictions is all about forecasting given an existing set of data. Let's start with the simplest kind of data, which is data in one dimension. Such data can be represented using points along a line such as a number line. For instance, let's say we took oil prices over a period of time and plotted along a line. Clearly though, representing such data along one dimension is restrictive because we can't express or even view relationships between different variables. So if you wanted to examine the

relationship seen between oil prices and government bond yields, that is interest rates in an economy, well then we will need to represent our data in all dimensions, that is as points in a plane. As soon as we do so, our eye starts to look for patterns, for some kind of code that can be drawn through all of these points. In fact, we can draw any number of curves to fix such data. For instance, we might draw a straight line or a squiggly line. A straight line drawn between two variables is set to represent a linear relationship. We could choose to make this curve pass through each point or in some sense try and fit the data using a representative curve. The more complex a relationship we are will to accept between these two variables, the more squiggly our curve get, the more closely it will match the data points that we have. Now a straight line representing a linear relationship is not squiggly at all, so you might think that the straight line is not good enough to affect our data, but in reality, creating a very complicated, very squiggly curve means that our curve may look or lead with new data points. This is a problem known as overfitting. Overfitting implies that we have created a curve which perfectly fits the data that we already have, but if we are given a new point, an out-of-sample point, well for this point, our curve, our squiggly curve might perform very badly indeed. For this reason, it's often sufficient and it's often very powerful to just go with the straight line between our data in two dimensions. The process of finding the best such line is described as linear regression. Measures of goodness of fit, such as the R-squared, which we'll discuss in great detail, help us to understand how strongly one variable explains another. If we set up the regression right, then if we get a high R-squared, indicating a high goodness of fit measure, it means that changes in oil prices are almost directly attributable for changes in government bond yields. And as we can see from the graph here, there is a strong linear relationship between the two variables and that's why we would in fact expect to have a very high R-squared close to the maximum possible value of one for the data on screen now. But on the other hand, if our data did not display an obvious visual relationship when graphed, you will expect to see a very low R-squared signifying a very poor quality of fit. This is show in the graph on screen now. The points are arranged seemingly at random on the graph. This is a tell-tale sign that the R-squared of a regression on such data will be low. Quantifying such cause-and-effect relationships is one of the most important applications of regression and the other equally important application is prediction. We use regression for prediction as follows. Given a set of points, we fit the line through it. Then given a new value of X, we use the line, that's a regression line, to predict the corresponding value of Y, and once more using regression, we can put a confidence interval on our own forecast. For instance, we can define an interval such that we have a 95% level of confidence that our prediction is going to lie within this range. All of this conversation so far was about data in all dimensions, particular and so that linear regression can just as easily be extended to an R _____ number of dimensions. If our

regression is carried on two-dimensional data, it's known as simple regression. If the data is in more than two dimensions, it is known as multiple regression. We will examine simple and multiple regression in more detail in the modules to come.

Reasons for Using Regression

Regression is an incredibly popular and powerful tool. Let's understand why. Regression is powerful. It's versatile because it can be used for all kinds of data, including non-linear relationships, and lastly, because it's deep. In fact, regression can be really thought of as the first crossover hit from machine learning that has gained wide acceptance in everyday life. We've already discussed the two most common applications of regression are in quantifying cause-and-effect relationships and in predicting given a set of existing data. For both of these use cases, regression is maybe the most popular tool out there because it naturally lends itself to both of those applications. Let's really quickly discuss each of them in turn starting with a discussion of using regression to explain variance. Let's say that we would like to examine or analyze a stock whose price has been rising lately. Is the stock rising because the market is rising as a whole or is there some stock-specific out performance, maybe good management in that particular company. Regression has played such an important role in analysis of this sort that even the names of these two phenomena come from regression. The first explanation is referred to as Beta. Beta is a financial issue which helps us understand how much a particular stock moves if the market as a whole moves by 1%. Explanation two is known as Alpha. This refers to that portion of the price rise which cannot be explained by the rise in the market as a whole. A stock which has a large positive value of Alpha is doing something right. The first step in using regression in a problem like this is to clearly identify the cause and the effect. It's important to identify cause-and-effect before running regression, otherwise we will end up with a nonsense regression. Coming back to this example, our explanatory variable here or the cause here is changes in the level of the market as a whole. If that's the cause, well then the effect is changes in the price of a particular stock. This is the dependent variable. Changes in the level of one particular stock can be attributed or can be explained using changes in the level as a market as a whole. Let's say the stock we are seeking to analyze is Google and let's say we that we can represent the market using a diversified stock index, such as the S&P 500. Then we can get a graph like the one you see on screen now. Returns of Google are represented on the Y axis and returns of the S&P 50 are represented on the X axis. And crucially, the regression line between these two sets of returns is represented by the equation, A + Bx, and that is a dead giveaway of where the terms Alpha and Beta came from. Alpha in this equation is the Y intercept. It is the point on the Y axis where our regression line

would cut that Y axis. You can see it represented by alpha on the screen and this is also the out performance of Google stock or the market as a whole. Beta, on the other hand, is the slope of this regression line. It can be interpreted as how much Google stock would move by for a unit change in the S&P 500. If the S&P 500 increases by 1%, then Google's stock will increase by Beta percent. Beta is the slope and it gives the sensitivity of Google stock with changes in the market as a whole. And in this way, we can interpret the magnitude and the sign of Alpha and Beta to analyze just how much of a recent price rise in Google stock is attributable to changes in the market as a whole. In fact, it turns out that there is an even more direct way of quantifying the cause-and-effect relationship using a goodness of fit measure called the R-square. We've already alluded to R-squared and we will discuss it in more detail in modules to come, but you can interpret the R-squared directly as the proportion of variation in the stocks returns that are explained by the market returns. That gives us a flavor of the first use case for linear regression. Let's now talk about the second common use case which is prediction. Let's say that interest rates are rising and commodity traders are worried. They know historically that when interest rates go up, the price oil tends to fall. But the key question is by how much will oil fall. This is key in order for them to decide whether to sell oil or not. Let's say that US government bond yields are presently at 2. 56% and traders feel that could rise to 3%. What they would like to know is if oil is currently trading at $50 a barrel, should they buy or sell oil. This is an extremely practical question traders have wrestled with such questions day in and day out. One way to solve this using linear regression is to plot oil prices on the Y axis and government bond yields on the X axis. Notice again that we have plotted the cause on the X axis, that's changes in government bond yields, and the effect on the Y axis, that is changes in oil prices. Now given a new value of X, that is for government bond yields equal to 3%, drop a perpendicular line onto our regression line, and from that regression line back onto the Y axis. The corresponding value on the Y axis is telling us that if government bond yields move to 3%, oil prices are going to fall from $50 at present to $43. And so, we can see intuitively how linear regression can be used for its two most common applications, explaining variance and making predictions.

Versatility of Regression

Another key reason to use regression is its versatility. You might think that linear regression is really is only useful when there is a linear or a straight-lined relationship between two data series, but this is not really the case because we can use any number of smart transformations to express two data series such that there is in fact a linear relationship between them. Let's make this a little more real using an example. It's pretty obvious just from eyeballing these two graphs

on screen that the relationship between X and Y is not linear, but that's not a deterrent. We can still use linear regression in such situations. In the case of exponential data, it is simply by taking logarithms. This will convert the relationship into a linear one. The dependent variable will now be log Y and the independent variable will still be X. We can adopt a similar experiment if the relationship is a polynomial one. Once again, we take logarithms. This will convert the relationship into a linear one between log of X and log of Y. There is another alternative in this situation which is to simply regress Y on the higher power of X, making this a linear relationship between Y and Cx-squared. We won't really go into the details of how these transformations will work, but you should know that using these transformations we can easily convert exponential or polynomial relationships into linear ones. In the case of the exponential relationship, we have a log of Y on the Y axis and X on the X axis and we carry out the regression as before. If the relationship is a polynomial one and if we decide to go the logarithm transformations route, we convert this into a linear relationship between log of Y and log of X. This, by the way, is one reason why returns of converting stock prices into stock returns is a favorite in really standard transformation applied before using regression. Mathematically, converting a slow-moving series, such as a stock price, into a faster moving series, such as its returns, this is somewhat equivalent to what you see on screen now. You'll have more to see on this in a little bit, but hopefully this is enough to give you a sense of the fact that employing smart transformations as a pre-processing step when there are non-linear relationships between data series is a really powerful way to extend the versatility of linear regression.

Regression as Machine Learning

The powerful and maybe the most interesting reason for you to use regression is because it's actually a surprisingly deep algorithm. In fact, think of linear regression as the first crossover hit from machine learning that became popular in mainstream business and other applications, and so if you gain an understanding of how regression works, you can then extend that understanding relatively easily into an understanding of other machine learning techniques, which are, of course, all the rage these days. Let's really quickly talk about what machine learning is. Let's consider one of the most common startup problems in machine learning that of classification. The question that we are trying to answer is are whales fish or mammals. Now of course, in reality, whales are mammals because they are members of the infraorder cetacean, so they are mammals by definition. But on the other hand, whales also share a bunch of similarities with fish. They look like fish, swim like fish, and move like fish. Let's say that we needed to automate the classification of animals into either fish or mammals. One way of doing this is to create a rule-based classifier. A

bunch of human experts will write a bunch of rules. One of those rules will be that if an animal belongs to the infraorder cetacean, that animal is a mammal. Now another way of creating a classifier will be the machine learning approach. Here we take a corpus of a large number of animals, both mammals and fish, and pass them or show them to a classification algorithm. In a sense, we would train that classification algorithm and the output of this process would be a classifier, a machine learning-based classifier, which can now be used to tell whether an animal is a fish or a mammal. Of course, it's incumbent on us to draw the attention of the classifier to the correct aspects of the corpus. If we tell our classifier that this animal breathes like a mammal and it gives birth like a mammal, we are telling it to focus on those attributes of the animal and it will then correctly classify a whale as a mammal. If we had focused on how the animal moves or where it lives, then the classification might have been incorrect. For us to get the correct classification, it's really important for us to correctly train our algorithm. And this little example really sums up the differences between ML-based and rule-based algorithms. ML-based systems tend to be dynamic changing the way in which they work, these are the data that they are trained on. Rule-based systems require trust and to be directively static. ML-based systems do not require a whole lot of heavy duty experts in a particular subject, rule-based system invariably do. Such experts can be really expensive and that's a win for the ML-based systems. On the other hand, a ML-based systems typically need to be trained using a corpus, the beta, the quality of this corpus, the more representative it is, and the larger it is, the better the ML-based classifier tends to do. This is in contrast with rule-based systems where the corpus is in a sense in the brains of the experts who create the rules. And this is why ML-based systems tend to require an explicit training step. Rule-based systems usually do not. If we now go back and look at how we use linear regression, we really would employing an ML-based algorithm. Let's see, for instance, how our analysis of Google stock was actually machine learning at work. Here we started out with a large number of data points representing returns bought for Google stock, as well as for the market as a whole via the S&P 500. These points together construed the corpus and we feed them into the regression algorithm, which is something that we will discuss shortly. The output of this regression algorithm is an ML-based predictor and that is a really simple predictor, it's just a line. It's a regression line with the equation y is equal to a + Bx. We can use this predictor to predict changes in the value of Google stock given changes in the level of the market as a whole. And this is how prediction using regression is a form of machine learning. We might plug in, for instance, a new data point, a new scenario that the market is going to go up by 10%. The question that we want to answer is how much will Google stock go up in response. And we're plugging in this value of 10%, and to our regression line, we can come up with an answer like Google is going to go up by 13%. If we change the scenario or the problem instance that we pass into our

predictor, we get a different forecast. For instance, if we predicted the market is going to go down by 10%, our regression-based predictor will tell us that Google is going to go down by a little bit more. It's going to go down by 13%. A lot of folks who use regression don't really know or don't really care that it's an example of machine learning. But for data professionals, this is an important point. An understanding of linear regression can easily be extended to a deeper understanding of other machine learning techniques as well. And this is why regression is such a great tool, as we've seen, it's powerful, versatile, and extremely deep.

Mean and Variance

Now that we've introduced regression as a way to connect the dots and we have discussed why it's so powerful and robust, let's also talk a little bit about some basic concepts from mathematics and statistics which will be useful to us as we move on. Let's start with the really simple concepts of mean and variance. Consider data in just one dimension. Could I please have your thoughtful, fact-based point-of-view on these numbers? Well, that point of view depends on the audience, of course. You would probably have one point of view for your boss who has a 5-second attention span, but a more detailed nuance version for your go-to colleague who has a 10-second attention span. Let's just start with the mean, which is the headline of any set of numbers. Given the set of points, X1 through Xn, the mean or the average is the one number that best represents all of these data points. And virtually, all of us know the definition and the formula for the mean, it's simply the average of a set of numbers. This one number is probably enough for us to answer our boss who has a very short attention span. But if we are talking to someone with a little more of an eye for details, the variation in our set of data is important as well. And there are host of statistics that attempt to capture this variation starting with the range which is simply the difference between the largest and the smallest of all of the numbers that we have. However, the range has some problems because it ignores the mean and its swayed, swayed far too much by outliers, and that's why we typically use a more robust measure of variation called the variance, that's the second important concept that we want to talk about. The variance is the second most important number if you are looking to summarize a set of numbers. Let's see how the variance is calculated. The variance starts from the mean deviation and the mean deviation in turn measures the distance of each point from the mean. So if you want to capture how much variation there is, one way of doing so is to square the mean deviation and then sum up all of those mean deviations and find their average and that, my friends, is exactly what the variance of a set of numbers is. Given a set of numbers, X1 through Xn, the variance is represented by the formula you see on screen now. The greater the dispersion of those points around the mean, the greater the variance is going to

be. There is one very small mathematical footnote, in practice our best estimate of the variance uses the denominator, n-1. Instead of the denominator n, this is known as Bessel's Correction. Do remember this formula, but you do need to pay attention to why Bessel's correction is required in the first place. You've probably also heard of the term standard deviation. The standard deviation is simply the square root of the variance of a given set of numbers. And so, the mean and the variance of a set of numbers succinctly summarize that set of numbers. You can form your thoughtful fact-based point of view very easily given these two numbers.

Probability Distributions and the Bell Curve

Let's understand the bell curve of the normal distribution using an example. Take a statement like this, Michael Jordan is a once-in-a-lifetime player. Let's try and represent a little more precisely what the statement is trying to get across. It's trying to say that Michael Jordan is an outlier in basketball ability. And so, if you could represent basketball ability on a number line, we'd have Michael Jordan way all on the extreme right. He would be an outlier. Notice also that there will also be another set of outliers that is the set of NBA players. These are already far more skilled than ordinary folks. A key insight here is that average levels of ability are very common. Very high or very low levels of ability are relatively infrequent. If we could block the number of people at different levels of skill, most ordinary folks would be clustered around that average level. In this graph, the NBA players would already be outliers and then Michael Jordan would be yet another further outlier to the right of the other NBA players. This diagram that you see on screen now tells us how common a specific level of skill is. And notice that the shape of this chart resembles a bell. This is a normal probability distribution, also known as a Gaussian curve or a normal distribution. If we have a lot of values which arrange themselves into a bell curve, well then, the average of those values is going to _____ really often. Very high or very low values are going to be unusual. Take 100 or 200 individuals and plot their heights or weights and you're almost certain to end up with a curve shaped like a bell curve. Now given a curve like this, we actually can answer a really interesting question. What is the probability of any specific value x occurring within the data that we've been given? And the answer to this lies in something known as the probability distribution function. There is a little bit of mathematical detail that I'm going to skip over here, but at a high level, you can think of a Probability Distribution Function called a PDF as a lookup table. It helps answer the question, given any value x, lower case x, how likely is that value to be found in the data? This likelihood of probability will be a number between 0 and 1. If we summed up all probabilities of all possible values, you would get to the number one and it turns out that bell shaped curve, which we saw a moment ago, is just on specific type of probability distribution

function. Given any value x and given a probability distribution function with mean, new, and standard deviation sigma, the probability of the value X occurring is that complicated mathematical formula on the right. You can safely ignore that formula for the remainder of this class. Just remember that normal distributions occur everywhere in nature and they have special properties which are relevant for linear regression. Let's focus on the mean and the standard deviation. As mentioned, we represent a normal distribution in terms of these two parameters, and from the shape of that bell curve, we can find what percentage of all values will lie within one or two or three standard deviations of the mean. For instance, for any normal distribution, for any bell curve, 68% of all data, that's about two-thirds of our data will lie within one standard deviation of the mean. That is in the interval from new - sigma from new + sigma. If we increase that interval to two standard deviations from the mean, then 95% of all values are going to lie within this range. And if we increase the interval to 3 sigma, well 99% of all values will lie within 3 standard deviations of the mean. This is in keeping with that intuition that average is common and that either very good or very bad is relatively rare. And so, using our newly-gained knowledge about normal distributions, we can interpret the statement, Michael Jordan is a once-in-a-lifetime player. If you took all of the basketball players that you are likely encounter during your lifetime and if you arrange them by ability, you get a bell curve, a normal distribution, and in that normal distribution, Michael Jordan will be five standard deviations away from the mean. He will be a mathematical freak, a once-in-a-lifetime player. Let's now tie back this knowledge about mean and variance to our discussion about regression. Recall that regression is an exercise in fitting a curve, in fitting a straight line through a set of points. Given a set of points, we are trying to find the equation of a line, y is equal to A + Bx such that every y coordinate can be perfectly expressed in terms of the corresponding X coordinate. This requires us to find the values of the constraints, A and B. Now although, we would ideally we probably cannot find any such constants, A and B, and so we've got to live with some small errors. Let's call those e1 through en. These error terms are actually more technically referred to as the residue of the regression and we've got to find the constants, A and B, such that the magnitude of the residue is in some sense minimized. And this is where just a little bit of statistical knowledge comes in handy because it turns out that in order to find the best fit regression line, we're going to need to make some assumptions about the regression residuals. These assumptions are properties, statistical properties that the residuals on the whole should satisfy. Where does these properties come from? Well from the theory of statics, which we can ignore for now. For instance, they ought to have 0 mean, have the same variance, be independent of each other, as well as of the independent variable, and they need to be normally distributed. These assumptions can be violated even when we are using regression, but we'll leave it there. This is what we will talk about

in the modules coming up next. But before we get there, let's quickly summarize all that we learned in this module now. We introduced regression as a way to connect the dots, literally, that is to fit a curve through a set of points. We understood the intuition behind the two main use cases for regression in quantifying cause-and-effect relationships and in prediction. We saw why regression is such a powerful, versatile, and robust tool. And we also learned how regression is an example of a machine learning technique. We then rounded off the module with a quick run-through of some concepts from statistics that we'd require as we delve deeper into regression.

Understanding Simple Regression Models

Setting up the Regression Problem

Hello, and welcome to this module on Understanding Simple Regression Models. As we've already mentioned, simple regression is the code that's used to described regression applied to two-dimensional data where we have one x variable and one y variable. In this module, we will set up the regression problem and describe its solution. We will then introduce simple regression models that have a single explanatory variable and we'll explain what that all means. Then we'll move on to using simple regression models for explaining variance and for making forecasts. And we'll wrap up by talking about the assumptions that underlay the regression model. These are assumptions that we are making whenever we use regression, and we have got to check after we've used regression whether any of these assumptions have been violated. So let's plunge in and let's start by talking about setting up the regression problem. Cause-and-effect, that's what regression is all about and we should only think about using regression when we clearly know what's the cause and what's the effect. The cause or the independent variable drives the effect which makes it a dependent variable. These terms are important. If X causes Y, X is the independent variable and Y is the dependent variable. These terms, independent and dependent variables, may be used again and again while discussing regression, so make sure that you remember them. X is also often referred to as an explanatory variable. Again, the terms, independent variable and explanatory variable are synonymous. Once we've clearly identified our cause and our effect variables, we can go ahead and plot them on a graph. And in doing so, we've got to be careful to plot the Y variable, that's the dependent variable, on the Y axis and the independent variable, or the explanatory variable, on the X axis. This is a part of the standard

terminology of regression. Independent variables or explanatory variables or X variables are always on the X axis and the dependent variables or the Y variables are always on the Y axis. And once we have these points set up nicely in a two-dimensional space that's a plane, the next step is to find the best fit line and this is the process of linear regression. And linear regression is the process of finding that best fit line that passes through these points and when we say best fit, we mean that in a very specific sense which I will describe in just a moment. These focus also on the term linear. That indicates that we are going to find a linear relationship or a straight-lined relationship between the X and the Y variables. Now let's compare two possible candidate lines. Line one has the equation A1 + B1X and Line 2 has the equation A2 + B2X. We can see both of these lines on screen now and almost instinctively we can tell that line 2 is not as good of fit as line 1 is. Let's quantify the condition and see how exactly the best fit line is found. If you remember a little bit of coordinate geometry, you will recall that the Y interceptor for line is given by the constant term in its equation and that implies that if we extend the first line, line one, until it meets the Y intercept, that Y intercept is going to be A1, exactly as you see on screen now. And we also know from coordinate geometry that the slope of a line, B1, gives us how much the Y variable is going to change by if the X variable changes by one unit. Here, because our lines are sloping downwards, if X increases by one unit, Y is going to decrease by B1 units. Again, the slope is a measure of the sensitivity of Y for changes in X. We can now interpret the coefficients of line two in exactly the same manner. The Y intercept of this line is A2 and the slope of this line is -B2 implying that if X increases by 1 unit, Y is going to decrease by B2 units.

The Best Fit Line

Okay, let's now move on and try and quantify our intuition that line one is a better fit. Let's take each line and drop vertical lines from each point onto the corresponding lines. Now if you were to add together the lines of all of these darker lines, this would give us a measure of how far from all of the points in aggregate our particular line is. And so, the greater, the sum of the lines of all of these dotted lines, the worse the fit a particular line is. So that's exactly what we'll do. We'll drop the dotted lines onto each of the candidate lines onto each possible regression line and calculate the sum of all of their lengths. With little footnote that while doing so, we take the sum of the squares of the lengths of those dotted lines and then the best fit line is the one where the sum of the squares of those dotted lines is minimum. This is an absolutely crucial insight. Those dotted lines in some sense represent the errors while fitting a curve to these points and that's why this particular method for finding the regression line is known as minimizing the least square error that is the object of off the regression problem. And in this way, we have defined the regression

problem, we have set up the regression problem as finding a line which minimizes an error function given a set of points. What is the error function? Well, that is the least square error. And what is the best fit line called? Well, it's called the regression line. It turns out that those dotted lines which represent the errors have an absolutely crucial role to play in starting or analyzing a regression. These are known as the residuals and the term residuals will come into play again and again in this course. The residuals of a regression are the difference between the actual and fitted values of the dependent variable. The term fitted value is also an important one. For any point, Xi, Yi, this is a point in the original data set, we can get a fitted value that is by dropping a perpendicular from that original point onto the regression line. We will have a lot more to see about residuals and about fitted values in the remainder of this course, so remember these terms. Now how do we know if we have mis-specified our regression problem. For instance, write the force with the line when actually a curve or a squiggly line is more appropriate. Well, we can tell this by examining the residuals. In a well-specified regression problem, the residuals should display a number of mathematical properties. They ought to have 0 mean. All of them are to have the same variance. These residuals should always be independent of each other, as well as off the original explanatory variable X, and ideally, this should also be normally distributed. If we carry out the regression problem and we find that any or all of these assumptions do not hold true in the residuals that we end up with, then it's a sign that we have done something wrong. We then are to go back and re-examine our regression model and possibly change it out quite substantially.

A Quick Taste of Regression in Excel

Now that we've set up the regression problem, solving the regression problem is actually quite a cookie cutter business. Solutions for the regression problem exist in Excel, R, Python, and pretty much every statistical package of programming language out there. This is really simple to do. Let's very quickly take a look at how easy it is to use regression. See in Microsoft Excel. In this really simple example, let's set up two sets of data which have a linear relationship within them. We start by defining a pair of constants A and B, and we assign the values of 10 and 5 to these 2 constants. Then, and this is where it gets somewhat interesting, we define our X and Y data series. Y is A + Bx and X is just a random number and we use the RANDBETWEEN function that you see there. In order to generate random numbers, we have specified are independent variable or our explanatory variable to be just a set of random numbers. You might notice that once I create all of those random numbers, I copy paste them as values because if I didn't do that, they would keep changing in value each time the sheet refreshes. So that takes care of setting up the X variable,

that's the predictor or the cause _____ variable. Let's now go ahead and set up the Y variable and this is where we make use of our constants A and B. Each value of Y is simply A + B * the corresponding value of X. Clearly, this is a linear or a straight-lined relationship by construction. We have explicitly chosen Y such that it is a linear function of X. Now that we've set up X and Y, we can go ahead and add out regression on them. As always, let's start by clearly labeling the cause and the effect. Here, X is the cause and Y is the effect. We now carry over the regression of Y on X. Now there are many different functions in Excel that allow us to do so. Here, I'm just going to make use of two of the simplest, these are the slope and intercept functions. Exactly as their names would suggest, the slope and the intercept functions return the values of the slope and the intercept from the regression line for a given set of X and Y. You can see that all that we need to pass in is the data series corresponding to Y and X in that order respectively. And here is what's cool. We can see that the results of our regression agree exactly with the initial constants, A and B, that we have used to set up the linear relationship and this is a test. This shows that the regression line in this case is the original line itself. Clearly, we had run regression on linear data on effect linear data, and so the slope and the intercept were exactly equal to the constants, A and B, that we had used to begin with. As a very quick side, we can also use the Excel function rsq to give us the value of the r-squared. This is a goodness fit measure which we shall talk in depth in the sections to come. In this example, because we have started over with effect linear data, the r-squared is as high as it can possibly be. It's at the periodical maximum of one. I will also use this as an opportunity to point out that regressing Y on X is not the same as regressing X on Y. If we reverse the order of Y and X in our function calls to the intercept and the slope functions, their values will change as well. Clearly, regressing X on Y is not the same as regressing Y on X and this is why it is so crucially important while using regression will have absolute clarity, to be absolutely certain beforehand about what the cause and the effect are. If one is not sure what the cause and the effect variables are, which direction the causation is going in, then we've got to go back and make sure that we understand our data better to begin with.

Solving the Regression Problem

We've discussed setting up the regression problem and we also carried out a really simple demo in Excel to show how most packages solve that regression problem really easily. Let's now turn from that solution and start sticking package on a spreadsheet program like Excel to the intuition underlying the solution. We defined the regression problem as finding the best fit line to a set of points. That best fit line is the one which minimizes the sum of the squares of the lengths of the errors and those errors of the residuals are the vertical distance between every point and the

corresponding fitted value on the regression line. The best such line, the line which minimizes the sum of the squares of the lengths of the residuals is called the regression line and finding this bet fit line is the objective of the regression problem. This is an optimization problem and we've got to find one line out of all possible lines that can be drawn to this set of points. We will not go into the mathematics of the statistics behind this, but you should know that there are no less than three different estimation methods for finding this line. For the case of simple regression involving one X variable and one Y variable, these three methods will all provide the same answer. The method of least squares is what we've discussed so far and it is this method which we will also use in more than two dimensions, so let's keep going with the intuition behind this method. Remember that we are seeking to find the line which minimizes the sum of the squares of all of the residuals. Those are the dotted lines on screen now. Let's assume that we found the best such line and that it has the equation, y is equal to A + Bx. Then the term A in this equation is the Y intercept, and B is the slope of the line giving us the relative change in Y for a one unit change in X. Let's start by labeling all of the points in our data set as X1, Y1 through Xn, Yn. If we now accumulated all of the X values, we would get the X vector consisting of the values X1 through to Xn. We can do the same with all of the Y values and that would give us a Y vector consisting of the values Y1 through Yn. Now for each point in our original data set, Xi, Yi, there is a corresponding point on our regression line and we can find this corresponding point by dropping a perpendicular, a vertical perpendicular from every point onto the regression line. By construction, this new point on the regression line will have the same X coordinate as the original point, but the Y coordinate will be different. Let's say that the corresponding Y coordinate is Yi' with a prime symbol just about as a super script. I figured this for every point in our data set. We could now collect all of the new values of Y into a vector called y' consisting of all of the fitted values. We now have both the original y values, as well as the new fitted y values. The original y, of course, is the dependent valuable, the new y' vector corresponds to the fitted values of the dependent variable. All of this terminology is important, so please remember what the fitted value is. And now, this helps us to precisely understand what the residual is. The residual for any point is simply the difference between the actual and the fitted values of the dependent variable.

Residuals and the Regression Assumptions

The idea of the residuals is an extremely important one. The residuals of a regression are the difference between the actual and the fitted values of the dependent variable. If the regression was a perfect fit, the residuals would all be equal to 0. That's what we saw in our little Excel demo. In general, the greater the lines of the residuals, the worse the quality of our regression. We

also saw in our little Excel demo that regressing y on x is not the same as regressing x on y and that's because when we regress y on x, we are seeking to minimize the residuals where those residuals are defined as the vertical dotted lines on screen now. And in contrast, when we regress x on y, we are actually seeking to minimize the sum of the squares of the horizontal errors. This is not the same as regressing y on x. It will give us a different regression line with a different equation. And so, once again, let me reiterate, you only should use regression when you are absolutely clear in your head about what is the cause and what is the effect. A limitation of regression models is that they cannot be used to find a cause-and-effect relationship. There are more powerful statistical methods, such as Granger Causality S, for instance, which can help with this regression inward. So far, we've only spoken about simple regression where our data can be represented in two dimensions, but most of the intuition that we have presented so far can easily be extended to multiple regression where we have multiple independent variables, but only one dependent variable. And as we've already discussed, there are different estimation methods, these are cookie cutter optimization techniques, for finding that best fit line, for finding the values of the regression coefficients, A and B. In the case of simple regression, these three estimation methods will yield the same result. More generally, the method of least squares is a good choice no matter what the dimensionality of your data because it has been proven that a least squares estimator has a set of desirable mathematical properties. It is the best linear unbiased estimator. Here, the word best is used to imply that the coefficients of a regression line calculated using the method of least squares have minimum variance. And the word unbiased means that the residuals have 0 mean, are uncorrelated to each other, and have equal variance. Recall, that we had already discussed some of the assumptions about how the residuals of a regression should behave statistically. Let's tie back this BLUE property of the minimum least squares estimator to the idealized properties of the residuals. This goes back to the regression assumptions which we had spoken about. Technically, these are assumptions about the regression error and not really about the regression residuals. Once again, my suggestion to you would be to ignore the mathematics behind this difference, as well as behind the reason for these assumptions. Let's just take them at face value for this course. The least squares method satisfies the first condition. It gives residuals which have 0 mean. Under the right mathematical conditions, it will also satisfy the condition of common variance where all of the residuals are drawn from the same distribution. Ideally, we would also want our residuals to be independent of each other, as well as off the explanatory variable X. The regression line opt in by the method of least squares scores on both of these aspects as well. There is some mathematical fine print around these which we need not get into here. Let's get to the last assumption. Ideally, we would want our residuals to be normally distributed, and this however, is not something that the method of least squares can guarantee

us, but four out of five isn't bad, and that is why regression using the method of least squares is such a popular and powerful technique. I should add that these properties on screen now are the idealized properties of residuals as we shall see in reality. We very often will use regression even when some or even most of these assumptions are not satisfied.

Explaining Variance Using Simple Regression

Now that we've understood the setup of the regression problem and also a glimpse to how it's solved by using the method of least squares, let's plunge into the two common applications of regression explaining variance and making predictions. Explaining variance is all about quantifying cause-and-effect relationships by making it explicit. How much variation in one data series is caused by variation in another data series. This is the application that we will focus on in this video. After this, we will talk about the second application, which is using regressions to make predictions extrapolating into new scenarios from data that we already have at hand. We have already alluded to the question of whether a stock was rising because of alpha or beta. How much of the increase in company X's stock is attributable to an increase in the market in the same period. Now it turns out that this is a problem which can be solved relatively easily using regression. In fact, we've already hinted at this, but it's also easy to really convert some blatant fallacies. And so, let's make sure that we understand some of these fallacies and how to avoid them. Let's start with something known as the post hoc fallacy. Let's say that a new roommate moves into your apartment and starting just about then, the weather turns gloomy. Now just because X happened before Y, it does not mean that X caused Y. In this case, it's likely that the new roommate moved in just around at the start of winter. And so, both of these factors coincidentally seem to happen around each other with one preceding the other. And so it will be a mistake, in fact, it will be the post hoc fallacy to say that the weather turned gloomy because your new roommate moved in. This one was relatively easy to spot. Let's turn to a more complex one demonstrating that correlation is not the same as causation. Let's say that we make two observations. The first is that the economy is booming. The country's GDP is growing faster than it was at any point in recent memory. And the second observation is that banks are lending freely. Not even Nobel prize winning economists seem to be able to agree on this one. Is the economy booming because banks are lending or are the banks lending because the economy is booming. Both ways of case can be made in either direction and that's why it is so important for me to emphasize, never ever use regression to establish a cause-and-effect relationship. Mentally, establish that relationship first and then use regression in order to quantify the extent of that cause-and-effect. This is actually a really important point because using regression is so easy that

a lot of folks end up using it when X happens before Y or merely even X and Y happen together. Neither of these is a good application or a good use case for regression to explain variation. You really only ought to use regression in the third case which is when X actually causes Y. The first is an example of post hoc fallacy. The second is an example of the correlation is causation fallacy. And only the third is genuine causation. We really only ought to use regression in the third case where we are sure that genuine causation already exists.

R-squared as Variance Explained

With that long warning out of the way, let's turn to quantifying the cause-and-effect relationship. Remember that we have the independent variable X. This is merely a vector consisting of all of the X coordinates of all of the points in our data set. This is the independent variable X. Similarly, we have a dependent variable vector that's Y. This is the vector of all of the Y coordinates of our data points. We pass this data through the regression cookie cutter using the method of least squares and we arrive at the regression line. This line is defined by its parameters, A and B, and it is the best fit line, it is the best linear unbiased estimator. Once we have the regression line, we can calculate the fitted values of the dependent variable by plugging in the corresponding values of X into our regression line equation. And we have already discussed the mathematical intuition behind this. The fitted value corresponds to the Y coordinate of the perpendicular drop from the original point onto the regression line. We collect all of these fitted values and this gives us a new vector Y′ consisting of all of the new Y coordinates and we can now calculate the residuals which are defined as the difference between the actual and the fitted values of the dependent variable. We can calculate one residual for every point. By collecting all of these residual lengths, we get a vector of residuals. Let's call this e and it's defined as y - y′. With just a little bit of mathematical jugglery, we get to an equation that expresses the variance of the dependent variable Y in terms of the variances of the fitted values, y′ and the residuals e. There is also a term in there called the covariance. We had not discussed covariance while talking about basic concepts and stats that we are going to need and that's because we actually do not need to know what covariance is all about. The covariance between these two terms always works out to be 0 and that's why this is going to be the first and last time we're going to talk about covariance in this course. This allows us to decompose the variance of the dependent variable into the variances of the fitted values and the residuals. Now, the variance of the dependent variable is known as the total variance or the total sum of squares, TSS. This is a measure of how volatile the dependent variable is in the first place. This is on the left-hand side of the decomposition, so let's plug it in and we can express ESS now as the sum of the variance of the fitted values and the variance of the residuals.

And it turns out that the variance of the fitted values is known as the explained variance or ESS. Remember that these fitted values come from the equation of the regression line and the better the fit of the regression line, the more closely ESS or explain variance will match with total variance or TSS. The difference between these two is the variance of the residuals that is the variance of E. This is known as the residual variance or the residual sum of squares RSS. This is a measure of just how much variance in the dependent variable could not be explained by the regression. So this variance of e can be written as the residual variance RSS, and so we can decompose the total variance TSS into the sum ESS + RSS. And clearly, the more variance we are able to explain, the more closely ESS will approach TSS, and this is quantified using a ratio called the R-squared. This is the percentage of total variance that has been explained by the regression. The higher the R-squared, the better the quality of the regression. This R-squared is always going to be a number between 0 and 1 or between 0 and 100%. And this really gets us to the heart of the matter. The R-squared tells us how much of the variation in one data series is caused by variation in another data series. The R-squared can easily be calculated in Excel or any other statistical software and it can easily be used to answer a question like how much of the increase in a stock's price is attributable or is explained by the rise in the market as a whole. We simply regress returns of the stock on the market returns and get the R-square of that regression. If the R-squared is very high, let's say in the range of 0. 8, then the stock's rise is mostly beta. It is mostly explained by the rise in the market. If the R-squared is very low, below 0. 3, then it's mostly alpha. It's because the company is actually doing something right.

Prediction Using Simple Regression

Let's now turn our attention to the second chief application of regression and that's prediction. As compared to using regression to explain variance, making predictions using regression models is actually conceptually really simple. And here are the steps in using regression to make predictions. We start with the data series, x and y. Once again, the key aspect here is knowing what's the cause and what's the effect. If required, we transform this data, for instance, by taking differences. Once that's done, we can specify the functional form, y is equal to A + Bx. We then pass this into the regression model and find the values of the parameters A and B. This can be done in Excel, or R, or Python, or most tools. We check the R-square and other statistics, such as the residuals. In the context of explaining variance, the R-square is the end result. That's what we really care about. Here, the R-square is not the end result, but it's still an important sign. The high R-squared tells us that our regression was a well-formed, well-specified model. If the R-squared is very low, we need to go back and check our input data or the functional form, but if it's high

enough, we can now use our regression model for predictions. Given a new value of X, what will be the corresponding value of Y. Well, simply plug the value of X into that regression model and estimate Y. We can then act on the basis of this prediction. Let's make all of this real using an example. Interest rates are rising, government bond yields are going up, and commodity traders are worried. What will oil prices become if US government bond yields rise to 3%? We simply project a vertical line from the X axis onto our regression line and then a horizontal line from that regression line onto the Y axis and that point of intersection will simply give us the price of oil as predicted by our regression model for the scenario where government yields have it as into 3%. You should also know that it is possible using regression to put a prediction interval. This is similar to a confidence interval around whatever prediction estimate we have come up with. This prediction interval is a range of values of the Y variable such that we have a 95% level of confidence that the actual value of Y will end up lying within this interval. We are not going to go into the nitty gritty of actually using prediction intervals because this course is intentionally light on the more statistical aspects of regression, but you should know that this is possible to do. You should know that it is possible to place a range around a particular estimate. And in addition, this range has the least error. When you are seeking a forecast, the value of Y that corresponds to the average value of X. That is the average X value of all of the points that we have. One last bit of intuition, the size of the confidence interval is related to the variance of the residuals. The smaller the variance of the residuals, the tighter the size of our prediction interval, and in other words, the more price that prediction is. Once again, this is only the intuition underlying prediction using regression. We will not go into the gory statistical details. And that in a nutshell is how we can use a regression model for the purposes of making predictions. Of course, the crucial final step, the crucial climax here is to actually act on that prediction. In this instance, we would presumably shred an oil future's contract called Express and make it a view on oil in some way. We might also construct a range of scenarios and take all of those scenarios into account before deciding how to act. We have more to see on actually building such predictors in Excel, in R, and in Python in the modules to come. But for now, let's pause and consolidate and quickly summarize all that we learned in this module. We started by setting up the regression problem and explaining how we could go about solving it. We understood the least-squares estimator and its BLUE property. This refers to the fact that the least squares estimator is the best linear unbiased estimator out there. There is some mathematical fine print, which is not very important to us. We then explored applying regression to both forecasting and explaining variance, making use of the R-square. And along the way, we also discussed the various assumptions about the residuals which are so important in determining that a regression model is a good one or not.

Implementing Simple Regression Models in Excel

Applying Simple Regression

Hello, and welcome to this module on Implementing Simple Regression in Excel. In this module, we will talk about the nitty gritty of actually of actually building regression models in Excel. Along the way, we will also understand and test the regression assumptions, again, using Excel functionality. We will apply regression to the two common use cases we've already discussed explaining variance and making forecasts. And to round off the module, we will learn how to avoid some common pitfalls in regression. So let's plunge in and let's start off learning how to applying simple regression. And remember that simple regression in our first tool, regression equations with one cause and one effect. The basic idea is given a set of points in a plane to find a line that is the best fit line through those points. Given a set of points with the coordinates, Xi, Yi, we can write the equations of these points. In terms of a regression line, our objective is to find the constants A and B ideally such that each of these equations is perfectly maintained. But in reality, it's usually impossible to find the constants A and B such that all of these equations are perfectly satisfied, and so we've got to learn to live with an except. A slight difference in each of these equations, this difference is called the residual. The residuals are the values E1 through En in the equations on screen now. The object of regression is to find the constants A and B such that the magnitude of residuals is minimized. So we find constants A and B which minimize the magnitude, in other words, they minimize the sum of the squares of these numbers. Such constants, A and B, give us the regression line. That's the best fit to all of these points. And then ideally, the residuals will satisfy a bunch of statistical properties. Ideally, they are to have 0 mean and they all should be drawn from a distribution with the same variance. They should be independent of each other, as well as off the original cause entry variable X. And lastly, ideally, the residuals should all be normally distributed. We've alluded to these assumptions in the past, but now we are going to dig down into them just a little bit. Let's start by focusing on three out of these five assumptions. The 0 mean, constant variance, and normal distribution assumptions can all be encapsulated into 1 single statement. Mathematically, all of the residuals are drawn from a normal distribution function with mean 0 and a constant variance, sigma-square. Of these 3 assumptions, the 0-mean assumption is always satisfied. This is by construction, that's because the least square's procedure finds the values of A and B in such a way that the residuals have mean 0. The implication of this is that the mean of the actual Y values will always equal the mean

of the fitted Y values. The other two parts of this assumption, that is the common variance and the normal distribution bits, these are harder to check, particularly in Excel, and so we usually indirectly check these by examining the appropriate scatter plots. More on this in just a moment. Let's move onto the next assumption made about residuals which is that they are independent of each other. Independence in statistical terms has a very specific definition. This is actually a very difficult assumption to test precisely, but we can use a shortcut to test this assumption using the Excel correlation function. What we are trying to do is measure independence, the independence of the residuals from themselves, but this is very hard to do, and so, we measure correlation instead and that's a lot easier to do. Let's understand this idea a little better. Correlation is a measure of linear similarity between two sets of numbers. For instance, if the 2 sets of numbers are effectively linear and that line slow upwards, the near correlation is going to be +1. As X increases, Y is also going to increase linearly or in proportion. Likewise, a correlation of -1 implies that as X increases, Y will decrease by the same constant proportion. And the third case is that of 0 correlation. This is where there is basically no linear relationship in the data. Changes in X can be assumed to be independent of changes in Y. And what we are trying to get at is that 0 correlation between 2 data series is often, but not always assigned that they are independent of each other. In the context of a regression, we want to check whether the residuals are independent of each other, and so we want to check whether the residuals have 0 correlation with themselves. But that poses a problem because for any data series, correlation with itself is always +1 and we get that on this using a simple little trick. We measure something known as the lag-1 autocorrelation instead. The autocorrelation of a series has its name would suggests is the correlation with itself. But in order to make sure that this does not work out to +1, we've got to shift our data a little bit. Given any series, we can shift the data as you can see on screen now and get two sub data sets and calculate the correlation between those two data sets. And that is exactly what we're going to do with the residuals of our regression. We are going to take those residuals and create a data series where we exclude 1 value and include the values 2 through n. This gives us one data series with n-1 items in it. We now go ahead and create another similar list, again of n10-1, but this time we exclude the last value rather than the first. This is another list. This also consists of all of the residuals and these 2 elements have n-2 elements in common. Now we use the correlation function to calculate the correlation of these 2 vectors, and ideally, this correlation should be 0. This method gives us a quick and easy way to ask for the self-independence of residuals in Excel. As we shall see, there is a more precise way of doing this in R. Moving on, the next assumption is that the residuals are also independent of the explanatory variable X. This assumption is simple enough to test. All we need to do in Excel is to plot a scatter plot with X on one axis and the residuals on the other axis. This scatter plot should give us no

discernable or clear pattern. If we were to fit a straight line through this, the R-square of that line should be as low as possible, and if that scatter plot of X versus the residuals that resembles the one on screen now, then all is well. The residuals are indeed independent of X.

Violations of Regression Assumptions

Violations of the Regression Assumptions are usually symptomatic of some underlying problem with the regression that we are seeking to run. Perhaps, there is just no cause-and-effect relationship between the variables. Here we would need to go back to the drawing board, understand our data, and come up with an alternative margin. Another possible issue is that is indeed a relationship, but we have mis-specified it. For instance, maybe the relationship is a non-linear one with an exponential or a polynomial curve being more appropriate than a linear one. Here, it can logarithms or transforming the data in some way might help. And lastly, it's possible that our model is incomplete. We have missed other explanatory variables and this will suggest that we need to shift to multiple regression. Let's cycle through these and try and fix them starting with the first problem. Our tell-tale sign that our variables just do not have a cause-and-effect relationship is a scatter plot of the Y versus the X variables. Ideally, we should see a nice linear shape like the one you see on screen now. This would indicate that the line drawn through these points is going to have a high R-square. However, it is also possible that our scatter plot of Y against X looks like this, a random pattern with no clear straight-lined relationship between Y and X. If you see a scatter plot like this, you are to abandon this model and go back to the data and come up with something better. Let's talk about the second possible category of problems. These come from mis-specified relationships. Recall that we have already discussed how the relationship between Y and X might be exponential or polynomial rather than linear. In such cases, the way to go is by taking logarithms or by taking returns. Either one of these smart transformations would give us linear data like that which you see on screen now. The term non-stationary data is used to refer to data items whose statistically properties change or time. Regression should only be carried out on stationary data. Smooth trending data like the two graphs on screen now are both examples of non-stationary data. We will not explain this term in more detail, but you should be aware that such data should never be used in regressions. The quality of the regression will always be poor. This is not a big problem. Once we are aware of it, we simply take first defenses of our non-stationary data using one of two methods, either the method of log differences or by converting to returns. Both of these methods are actually mathematically quite closely related to each other. Both of them involve transforming our original smooth data series into a less more data series of the differences between adjacent points and

then regressing the new transform data series. The manner in which these differences are taken varies. In this course, we are going to stick with the method of returns where we will transform each data series into a new data series of the percentage changes. You can safely do this pretty much any time you come across a non-linear relationship in your data. But the question that arises then is how do we know whether the relationship is linear or not. One tell-tale sign is often an excessively high R-squared. If you find that in your regression you have an R-square of greater than 90%, then you should go back and check your residuals because it's very possible that you have performed a regression on a mis-specified model. In just a little bit, we are going to perform a regression of smoothly trending data of the places of Google stock versus the NASDAQ and we will end up with a linear relationship that looks just about perfect. As we shall see in the Excel demo coming up, this use an R-squared of almost 93%. However, this is actually a mis-specified regression because the residuals display a high level of autocorrelation. And what's worse, there isn't really a reliable way of detecting this in Excel and this will be one of the strong arguments in favor of using R for regressions. But this is not a fail-safe method because sometimes you do end up with legitimate regressions which have a high R-square. The real way to know this with certainty is by testing whether the residuals are independent of each other and that gets us back to the conversation we had about testing the lag-1 autocorrelation of the residuals. Remember that independence is hard to quantify, and so we are using correlation as a shortcut instead, but we can't use correlation with self because for any data series, correlation with self is always going to be 1. We will, therefore, use the lag-1 autocorrelation. This is a procedure that we had discussed in some detail. As we shall see, there is a more precise way of doing this in R. The third common problem with simple regression is that of incompletely specified relationships. Here typically, we are missing some X variables. We are going with simple regression that has one cause and one effect, when in reality, we need to add more X variables. How do you know if this is the case? It turns out that the answer to this lies in a scatter plot of the X values versus the residuals. If the residuals are independent of X, then chances are that we've captured most of the explanatory variables that does most of the causes. However, on the other hand, if we find that there is some clear pattern which can be drawn between X and the residuals, then most likely we have missed out on some underlying cause of variation in the Y variable. A bad residual plot like the one we just saw indicates that some of the fundamental assumptions of the regression have been violated. For instance, it can place the residuals are not drawn from a normal distribution with constant variance. This phenomenon is referred to as heteroskedasticity. Hetero meaning different and skedasticity referring to the variance of the residuals. Heteroskedasticity has many possible causes. The most likely one in our context is that we have missed out on some explanatory variables. We need to go back to our data and understand it more thoroughly, and potentially,

switch from a simple regression to a multiple regression model. So in summary, we have discussed three risks with simple regression. The first is that there is no cause-and-effect relationship between our X and Y variables. The second is that there is a relationship, but we have mis-specified it. And the third is that we have specified an incomplete relationship. We have also seen that we can diagnose all of these three using a combination of the R-square, a scatter plot of X against Y, and a scatter plot of X against the residuals of the regression. Having diagnosed these risks, how do we go about mitigating them? If there is no cause-and-effect relationship, we've got little choice, but to head back to the drawing board. If there is a mis-specified relationship, the safest bet is to transform our data either by taking logarithms or converting from prices into returns. And we have specified an incomplete relationship, then we are best off switching from simple regression to multiple regression taking into account more X variables in our model.

Fast Prototyping in Excel

Let's get our hands dirty. Let's now plunge into a series of quick demos using regression. In this demo, we will start by downloading data from Yahoo Finance and then in one step see how Excel allows us to find the regression coefficients and the R-square. Yahoo Finance is a great source for financial data. Here we are going to download data for a bunch of symbols. Let's start with Google whose stock ticker is GOOG, that's G-OO-G. There's no real rocket science to getting the data from Yahoo Finance. We scroll down to the bottom of the page, we click on the link saying download the spreadsheet, it gives us a file, which we rename so that it's descriptive and we are good to go. We then go ahead and repeat this process for Exxon Mobil. In a later example, we will use oil prices to expand some of the radiation in Exxon stock. So we go ahead and download the data for an important oil EPF what they call USO. And then the same deal for two important equity industries, the S&P 500 and then the tech heavy NASDAQ composite. And so we've downloaded a whole bunch of csv files. These csv files are all safely on our local machine and we renamed each of these to have a descriptive name. We can now go ahead and open a bunch of these csv files and that's exactly what we'll do. I'm going to open the files for Google, NASDAQ, Exxon and copy all of them into the same workbook. This workbook is now ready for us to use. Let's just copy over the column titled adjusted close from the worksheets for Google, Exxon, and the NASDAQ into one single worksheet so that we can use it to set up our regression. Let's move now to step two of our demo which is the use of regression plots in Excel. Excel is maybe the best programming tool out there. It's just incredible how easy Excel makes it for folks who try out almost anything. In order to try out regression, all we need to do is select our data. So we select

our X and Y data in this instance, just the prices of NASDAQ and Google. We hit insert and there we look for the scatter plot, and from that we select a format, we click on the dots there, a menu opens up, in that menu we ask that Excel include the regression coefficients, as well as the R-square, and we have our regression equation right there on screen now. And so, with just a couple of clicks of the mouse, we've got a scatter plot, a fitted trend line, the values of alpha and beta, and also the R-square. Excel has pretty much done it all. Look at the R-square and notice something about it. It is extremely high, an R-square of 0. 93, and this ought to set some alarm bells ringing. Recall our conversation about R-squares that are too high. This is a sign that our regression is mis-specified. And sure enough, we actually ought to be running this regression not on the prices of Google and the NASDAQ, but rather using returns. How do we know that we've got to convert these prices to returns? Well because these are non-stationary data series, and as we discussed, such data series need to be transformed either by using log differences or returns. In this example, we'll go with the returns approach. Folks who are dealing with financial data series tend to already be intuitively thinking in terms of returns, in terms of percentage changes and that's why in most financial applications, we carry out regressions based on the returns of time series, that is by percentage changes in the prices of Google, the NASDAQ, and so on. Let's just copy over that formula for the percentage changes and populate the return series for our three data sets. And in order to calculate these returns, we've got to be sure to line up our dates from smallest to biggest. Once that's done, we can go ahead and calculate the regression coefficients and the R-square just as before. And this time, when we create a scatter plot of Y against X, we get a much more reasonable looking curve. We add the trend line and the equation and we see that we've gotten an R-square of 0. 38. The value of the intercept A or alpha is 0. 5% and the value of beta is 1. 06. This is a legitimate regression, and in this way, we have gotten Excel to carry out the regression for us with just a few clicks of the mouse. Also, keep in mind the significance of the intercept and the slope of the regression line. Here the value of E, which is 0. 5% represent the outperformance of Google or the NASDAQ index. Here within the sample, Google is displaying an alpha of 0. 5% per month. That is a significant outperformance for the stock. It indicates that Google's management is doing something right. And likewise, we can interpret the value of the beta that's the slope coefficient, 1. 06, has Google sensitivity, but changes in the NASDAQ. If the NASDAQ changes by 1%, say an increase of 1%, Google is likely to increase by a little bit more, by 1. 06%. And because the R-square is 38. 6%, that is the proportion of variation in the returns of Google stock that are explained by the NASDAQ alone.

Slope, Intercept, and Prediction

Let's now move onto the next step of our demo which involves finding the regression coefficients using two Excel functions named Slope and Intercept respectively. It's useful to know how to do this and that's because the slope and the intercept are now available to you in your worksheet for use for forecasting or for other calculations. These formally are about as simple as you can imagine. You simply specify the values of your known Y values and of your known X values in that order. Remember that we have already taken a quick look at how slope and intercept work in an earlier module. Let's now move on to maybe the most interesting part of a regression analysis which is sanity checking the residuals and this is where we are going to make use of the slope and the intercept which we calculated just a moment ago. The residuals of a regression are simply the difference between the actual and the fitted values of Y. And where do the fitted values of Y come from? Well from the regression line, of course. And so, we will calculate the fitted value of Y by plugging in the corresponding value of X into the regression line using the slope and the intercept. In this way, we get a vector of all of the fitted values of Y, we then calculate the residuals as the difference between the actual and loose fitted values. Now that we have all of the residuals, we are all set. We can go ahead and study them and make sure that the regression assumptions are not violated. Let's start with the 0 mean assumption. We simply take the average of all of the residuals of our regression and that works out to be 0. As we've already discussed, this is by construction. This assumption will always hold true when we are using a least-squares regression. Next up, we calculate the variance of the residuals using the VAR function. This is not something that we are really going to use at least for now. We'll come back to it when we talk about multiple regression in R. And now onto checking whether the residuals are independent of themselves. As discussed before, we are going to take two subsets of all of the residuals. The first excludes only the first value and includes the values 2 through n. Another subset will include the values 1 to n-1 and exclude only the last value. And then, now that we have 2 subsets of the same len, we can use the correlation function CORREL in Excel to test that these 2 vectors have correlation 0. Ideally, this correlation should be 0. This is equivalent to seeing that the lag-1 autocorrelation of the residuals is 0 and this is a very good indication that the residuals are independent of themselves. And here the result is almost 0. That's a good sign. That correlation of -0. 03 basically tells us that the residuals are indeed independent of themselves. So far so good. Let's now go ahead and plot scatter plot of the residuals versus the X variable that returns in the NASDAQ. And here, once again, we create the scatter plot, we go ahead and add a trend line, and the regression equation, and we note with relief that there is no clear or discernable pattern. This implies that the regression residuals are independent of the X variable. And now that we've satisfied ourselves and the residuals of our regression are okay, we can go ahead and use that regression in order to do some forecasting. Remember now the intuition behind how we use

regression for prediction. We find the regression line, which is the best fit line. Then given a new value of X, such as a 5% rise in the NASDAQ, we project that onto the regression line and then onto the Y axis to find the corresponding inquiries in the value of Google stock. So we go ahead and create a set of scenarios corresponding to changes in the NASDAQ ranging from a fall of 10% through to a rise of 10%. We then go ahead and use the forecast function in Excel in order to find the corresponding values of Google returns. The forecast function in Excel is really simple. It takes in a new value of X, known values of Y, and known values of X, and it returns the new, that is the forecast value of Y. And in this way, we are able to forecast all of the changes in Google stock corresponding to the scenarios that we have outlined. An equivalent way of getting these forecasts is to plug the corresponding values of X into our regression line. We could simply multiply each new scenario value of X by our slope, add the intercept, and we will get exactly the same answer as the forecast function in Excel. So that's what we do. We recalculate all of the forecasts manually using the regression line and we take the difference between these 2 to satisfy ourselves that it is 0 in every event.

Poorly Specified Regression Models

Now that we saw one example of a good regression model which satisfied all of the regression assumptions, let's also see what a couple of badly specified regression models might look like. Let's simply make a copy of the worksheet containing our good regression model and use it to carry out a pretty meaningless regression. Here we are going to regress Google's returns and Exxon returns. This really doesn't make any sense at all. It's unlikely that changes in Exxon are going to be a cause for changes in Google stock. And sure enough, when we carry out this meaningless regression, we find that we get a very low R-square of just 0. 09. We've already seen that we should be suspicious of very high R-squares which I see higher than 90%. Equally, we should be suspicious of very low R-squares say below 20% because it's very likely that we have simply misunderstood the cause-and-effect relationships in our data, and this is also confirmed by the shape of the scatter plot. Between Y and X, you can see there isn't a very strong linear relationship visible there. Almost everyone who has ever carried out a regression knows that a low R-square is a sign of trouble, but there are plenty of folks who carry out regressions on non-stationary data. And so, let's run one such regression. This is a mis-specified regression which just regresses the prices of Google on the prices of the NASDAQ. Remember, this is the scatter plot that we had examined which gave us an R-square of more than 90%, 93% to be precise. This kind of a regression is really dangerous because it looks amazing. Folks carrying out such regressions think that they are onto something really cool. But the problem is that when they sanity check

their residuals, they are likely to find that high R-square is accompanied by a very high level of autocorrelation in the residuals. Let's confirm that here. We have an R-square of 93%. Let's now find the lag-1 autocorrelation of our residuals and that works out to be 0. 88. A correlation that is so close to the maximum possible value of +1 is a clear sign that the regression assumptions have been violated. Now it would have been awesome to actually nail down the precise extent of that autocorrelation of the residuals, but it turns out that this is virtually impossible to do easily in Excel. On the other hand, it's trivial to do this in R as we should demonstrate when we talk about regression in R coming up next. And those two quick examples of the misuse of regression around our _____ conversation of simple regression in this module. There are a couple of topics that we have not spoken about, notably, the LINEST function in Excel and the concept of E-STATS of the regression parameters. We will address both of these while discussing multiple regression in Excel in a module that's coming up ahead. But first, let's summarize all that we've learned so far. We covered a lot of ground in this module. We learned how to build Excel regression models and avoid some of the common pitfalls associated with regression with R-squares that are either too high or too low. And we learned a solid grasp of the two most common applications of regression that is explaining variance and making forecasts.

Implementing Simple Regression Models in R

R for Regression

Hello, and welcome to this module on Implementing Simple Regression Models in R. In this module, we are going to pretty much repeat everything that we just did in Excel, but this time our choice of tool will be different. So we will build our regression models in R. We will test the regression assumptions and correct their action if they are violated and we will carry out the two common use cases of regression to explain variance and to make forecasts. And along the way, we will also acquire enough knowledge to avoid some of the common pitfalls of regression that folks run into. Switching from Excel to R takes a little bit of getting used to because Excel is basically the fastest and easiest prototyping tool out there. If you were to take all of the possible technologies in which we could implement regression models, Excel would be an outlier or on the far right in terms of ease of prototyping. But that ease of prototyping comes with a cost. Excel is also an unfortunate outlier in its robustness and reuse and I mean this in a bad way. Robustness

and reuse are not really Excel's strong points. Now Excel, of course, is an extremely popular and successful piece of software and its really well written. It adheres to this design principle. It makes its common use case easy while also ensuring that the difficult use cases are still possible and this is an accurate statement about regression in Excel. It is not the common use case for Excel users. In fact, if we were to compare the three tools that we are discussing in this course Excel, R, and Python, Excel and Python are both general purpose technologies. R is quite specialized to applications like regression. For instance, a common use case for regression in Excel would be a person trying to create one slide summarizing a regression analysis for an important presentation. The corresponding use case in R would be a case study on regression that has to be presented at the seminar. And a similar use case for Python might be a trading model which scrapes websites, finds out which stocks are trending, and then combines trading signals that implement sentiment analysis and regression, and in a sense, these demonstrate the great strengths of these tools. Excel is a perfect complement for use in presentations. R has a reputation for being heavily used in academia and in research. And Python is heavily used in industry and by very hands-on practitioners. All of this is true to a certain extent, but as far as the specific use case of regression goes, my suggestion to you is that you adopt R for any kind of regression corresponding to any of these three use cases. R has the visualization capabilities that you need to make nifty graphics for your presentation. It allows you to dig deep into your data. And it also allows you to implement a variety of techniques. Use R for regression. It really makes sense no matter what your use case is. We've yet to really discuss Python at all, so let's confine our conversation for now to a comparison between Excel and R. The two big factors in your choice of tool are to be the number of independent variables and whether you really want to sanity check your regression model thoroughly or whether you prefer to just accept it as it is. Let's set up a really simple two by two with these two parameters and cycle through the quadrants one by one. If you plan to carry out simple regression and you don't really care about sanity checking your result too strongly, then Excel is a good choice for you. As we just saw, Excel is a great prototyping tool. You can get your slope, intercept, and r-squared with just a few clicks of a mouse. This was perhaps the simplest of the four quadrants. Let's shift to the most complicated which is where you have multiple regression and you also wish to really dig deep into your model. In this use case, it's a no brainer, you definitely should be using R. We'll have more to see on this when we talk about multiple regression, but in a nutshell, Excel just doesn't have the necessary support. It doesn't even support the adjusted R-square, for instance. Moving on, what if you'd like to run a simple regression or you'd like to go pretty deep into its results. Again, in this case, R is your tool of choice because it allows you to robustly measure properties such as independence of the residuals. And the last quadrant describes the situation where you are going to perform a

multiple regression, but you are not going to dig deep into its results. Let me assure you first of all that this is a dangerous undertaking. Unlike simple regressions, multiple regressions can go badly astray if you are not quite sure if you have set them up correctly. But in any case, even in this situation, R should be your tool of choice. Excel just does not have even a hygiene level of support for multiple regression. So that's a quick comparison of Excel and R, but once again, my suggestion to you is to use R for regression no matter what your use case.

Data Frames in R

Let's plunge right away into a set of demos that show how simple regression models are built in R. I'm going to make use of the simple R GUI. This is extremely simple to download and install both in Windows and in UNIX-like machines. You also can use a tool known as R-Studio which is something like an IDE for R. Let's really quickly run through that installation for Windows. As you can see, I start by Googling R for Windows. I click on the very first link that shows up. This leads us to the appropriate page on the cran. r-project. org website. I go ahead and download the appropriate executable. Now we click and run that executable file. This installs R on my machine and places a shortcut to the R GUI on my desktop. We go ahead and double-click on that desktop icon and that opens up the R GUI and we are in business. Let's go ahead and maximize the console window and get started by defining a variable. As you can see, it's extremely simple to get up and running with R and hopefully this course will have given you enough of a flavor of R to peek your interest in this topic. There are a host of excellent courses on Pluralsight on this topic and these are a great starting point for your exploration of the capabilities of R. We are basically going to reimplement all of the regressions that we just carried out in the model on Excel, and so we make use of the same csv files with the raw data for Google, Exxon, and other financial assets that we have obtained from Yahoo Finance. R has excellent functionality for dealing with csv files, and so let's plunge right in. As you can see, I have created a variable called nasdaqFile and I have stored in this variable the part to the corresponding data file. I then go ahead and repeat this for a set of files. Next up, let's make use of R's functionality that allows us to read in the contents of a file into a data frame. Now our data frame is one of many data representations in R. A data frame can be thought of as data arranged in rows and columns where the columns have the same lengths and also have names. Notice here that we are instructing R while reading in the file to make use of a header. We specify that the header is equal to true. We also specify that the separator is a comma. Notice also the bit between the square brackets. We are explicitly telling R that we are only interested in two columns to be read in from that csv file and those two columns have the names Date and Adj. Close. This is a way of telling R that however many columns you

find in that data file, please ignore all other than the ones with these two names in the header. And because R has an interactive shell, we can explore our data. We simply type out the name of our data frame and hit Enter and the results flash before our eyes. As you can see, this consists of the date and the adjusted close. Notice how the data frame has 132 rules. Remember also that the rules have names and we can explore those names by using the function names. We can see here that the names are indeed Date and Adj. Close. Let's go ahead and change the name of the second column to goog. price. Let's now go ahead and repeat those operations, this time for the NASDAQ. So we invoke the read. table command once again. This will read in all of the data that we have for the NASDAQ into a data frame called NASDAQ. We will then change the name of the second column to nasdaq. price. This is a nice descriptive name so that we know what asset we are referring to. Now that we have two data frames, one with data for Google, and the other with data for the NASDAQ, let's go ahead and merge those two data frames. Let's also make sure that we sort our data so that it is an decreasing order of date. This will become important when we want to calculate returns. Notice how we use the merge function which takes in a couple of data frames, as well as a column name, that parameter, the buy parameter, can part of as the join key. In fact, it's no coincidence that the semantics of this merge command are similar to SQL joins. The data representations in R are very rich and really come in handy for data like this. After we merge the two data frames, but before we sort the combined data frame on the date column, we are careful to make sure that date column has the correct type. The command as. Date is basically telling R to treat the contents of that column as if they are dates. There is some complexity to the handling or to the _____ of columns and data frames which we need not get into for now. Once that's done, we take advantage of the interactive shell to explore our data and notice how our data is now arranged in decreasing order of date. This ordering is important because, as we've already discussed, we are going to convert this data into returns in just a little bit.

Regressing Returns, Not Prices

Now that we have all of our data nicely arranged in one data frame, let's go ahead and plot some of the data. Let's invoke the plot command, along with a data frame name and a column name. Notice in particular how we are able to refer to a specific column of our data frame by using the $ operator. And because we have just sorted this data with dates and decreasing order, this plot actually has the most recent data points way over on the left. So it looks as if Google's price has been falling over the past 10 years, whereas in reality, of course, that price has gone up quite a bit. Now we've already discussed how important it is to only perform regressions on stationary data. Let's mix things up and let's carry out a badly specified regression. We are now going to use the

lm command to build a linear model which regresses Google's price on the NASDAQ's price. Lm is the work horse regression function in R. It returns a model object which we have stored. Let's now plot a scatter plot of the X and the Y variables. Once again, we invoke the plot command, but this time we pass in two data series, goog. price and Nasdaq. price and the reserves are displayed in the pane over on the right. As you can see, this is a very perfect scatter plot indicating a very high R-square, and as we have already discussed, such perfect in reality in this scatter plot is probably a sign of trouble. And now to view the results of our regression, in detail we call the function summary and pass in as a parameter the googMisSpecified linear model object. There is a lot of information here, some of which we will not discuss in this module. Let's hone in on the important bits. Notice that the R-square is 0. 9276. This matches perfectly with the corresponding R-square we got in Excel. Notice also that the intercept is -115 that's in scientific notation and the slope is 0. 156. This regression also very helpfully tells us how statistically significant these parameters are. Notice the three asterisks marks over on the right for each of the parameters. Also notice right up top how we get some of the information about the residuals, their minimum and maximum values, as well as the median. This summary does not contain the mean because, of course, the mean of residuals in a regression is always equal to 0 by construction. For now, let's not talk about some of the other information that comes up here. Let's not talk about the standard error, or the t value, or the p value, or the multiple R-square, or the F statistic. We'll discuss all of these when we are talking about multiple regression. Let's examine a scatter plot of the residuals plotted against X that's the NASDAQ. That residual plot doesn't really signal that there is anything particularly wrong. It's not easy to pick out any pattern. But here is where it gets really interesting. R has a function called acf and this returns all of the autocorrelations for whatever time series you invoke it on. And this now is a dead giveaway. Remember our conversation about how autocorrelations are an excellent way of testing whether the residuals of a regression are independent or not. In this case, all of the autocorrelations for lags 1 through 20 are reported and you can see that the lag-1 autocorrelation is at approximately 0. 9. It's somewhere between 0. 8 and 1. Recall that in the Excel regression we had calculated only this one autocorrelation and we had gotten a value of 0. 88. This confirms our suspicion that this is a badly specified regression. Notice how effortlessly R was able to provide us with all of this rich information. This would have helped us to spot autocorrelation even if it is was present at lags other than one. In Excel, the way we have set it up, we will only had detected lag-1 autocorrelation. And from this, we can conclude that the regression model we had specified was indeed flawed. We need to fix this, so we need to calculate returns, and that's exactly what we do on screen now. This statement is where all the action is, so let's break it down. First note, any time you see a negative index while sub setting a data frame, that refers to some index, which could be

a rule number or a column number, that has to be excluded from whatever calculation we are doing. Now our data frame can be accessed using both row and column indices. And so when we have two negative indices, the first negative index tells us which row to exclude and the second negative index tells us which column to exclude and we need to exclude one row from each of our two sub matrices because we are going to calculate returns by taking the proportion of two rows element-wise. Each of our two sub matrices will have one less row than the original. We exclude the top row in one and the bottom row in the other. We do not want to be taking the returns of our dates and that's why throughout this formula we are excluding the column one. This is what we do using the index -1. The other bit worth keeping in mind is that if we perform operations on vectors of the same size, they are carried out element-wise. So we are effectively creating two matrices. The first of these two matrices excludes only the last row from the original data and the second of those matrices excludes only the first row, and in this way, we get two matrices of the same size with all of the rows having their dates offset by just one. And then taking the ratio of one to the other and subtracting one from that. This, of course, is exactly how we calculate returns. So in summary, what we are doing here is taking all of the columns of our data, other than the date column, and then creating two versions of this in which we exclude the first and the last rows respectively, taking the division of these two matrices element-wise and subtracting one from that. This gives us the percentage returns for every row in our data and that still leaves the problematic issue of one additional planned row to go at the end and we get rid of that last row by stripping it out of the data entirely in the next step. The result is we now have a data frame which has one less row than the original data frame and inside which all of the values are percentage returns. Let's also name these columns appropriately. We named them goog. returns and nasdaq. returns. And as before, we make use of the interactive shell to explore our data. Just type out the name of the variable goog, hit Enter, and we see all of the data before us. Notice that our data now ends one month later than it did previously. This happens any time you convert prices into returns because of course you don't really have any return for the first month in your original price data series.

Prediction with Regression in R

Now that we have all of the data that we want converted to returns and available in our data frame, let's invoke the eliminator and let's specify the functional form that we want. We are regressing goog. returns or nasdaq. returns. You get back a linear model object. We invoke the summary function on this object and the results of our regression are in front of us. Once again, these results exactly match the results of our corresponding Excel regression. The r-square is 0.

386 and the values of the slope and the intercept agree as well. Once again, we will gloss over the wealth of information that is reported. More on that when we discuss multiple regression. Let's keep going and build a scatter plot of the X and the Y variables. Here, both sets of data represent returns and you see a nice linear relationship over in the pane on the right. This is the linear relationship corresponding to an r-square of 0. 386. Let's for completeness also plot a scatter plot of the residuals versus the X variables. This is a good-looking scatter plot. We don't see any discernable pattern. That's great. We also use the acf function and all of the autocorrelations of our residuals are basically 0. Notice the dashed lines. Those show statistical significance and virtual none of our residuals are statistically significant. This is a certificate of good health for our regression, and so we can move on. Now that we are satisfied that our regression model is a good one, we can now make use of it to forecast new values of X and Y. Let's create a set of scenarios corresponding to returns in the NASDAQ. These represent the new values of the X variable and we do so by using R's facility available for creating sequences. We specify an initial value that is -0. 1, a final value that is +0. 1, and a step size of 0. 01 or 1%. This we then store inside our data frame with the column name nasdaq. returns. When we explore what we just have, we find that we have a nice set of scenarios corresponding to changes in the NASDAQ ranging from -10% up to +10% in increments of 1%. You might notice that we have renamed this data frame goog and we have set up the exact column name that we had in the data frame using the regression and there is a reason for this. This is a quirk of the R forecast function. The data frame that is passed into R's forecast function must resemble in its name and column name the original data frame used for the model. Then given a new value of X such as a 5% rise in the NASDAQ, we project that onto the regression line and then onto the Y axis to find the corresponding inquiries in the value of Google stock. And we invoke the predict function. We need to pass in at least two parameters. The first of these is the linear model and the second of these is the data frame with the new values and the resulting output contains all of the forecast Y values and these agree perfectly with the forecast function results that we adopted in Excel. Hopefully, this demonstration so far has already shown how much richer R's support for regression is. Let's also now see how R supports code reuse. Let's take all of the preprocessing that we've performed so far and collect it into a function. Let's call this function preProcess. All we need to specify are the names of the files containing data for the X and the Y variables. Everything else is done automatically and what we get back is a data frame with the return series that we can use directly to carry out the regression. And now we can see the power of R's regression really coming into its own. We invoke our function passing in the names of the files for the NASDAQ and for Google and with one line of code we have our regression results. Notice how the NASDAQ explains 38% of the variation in Google's returns and how Google's beta is 1. 06. This is the same regression that we

had performed just a moment ago, but we have done it now using a nice little function. Let's also carry out another mis-specified regression or rather a meaningless regression. Let us regress Google on Exxon. And here once again, one line of code is enough for us to get back our regression results. You see that the r-square is very low. It's just 9%. This is a hint to us that this is not a good regression model. And this call to the eliminator on screen now also indicates another retraction of regression in R. We can specify how we would like to deal with missing data. Notice that we specify that the action for data that is not available is to omit the data. We _____ down a little more with missing data when we get to multiple regression, but that gets us to the end of this demo on simple regression models in R. Let's quickly summarize all that we learned. Just like with Excel, we learned how to build regression models in R and we also understood the deferring use cases for different tools such as Excel, Python, and R. We learned how to avoid some of the common traps that folks fall into while trying to carry out regressions. And we covered the basic use cases of regression, that is explaining variance and making forecasts.

Implementing Simple Regression Models in Python

When Python Is the Right Tool for Regression?

Hello, and welcome to this module on Implementing Simple Regression Models in Python. In this module, we're going to cover much of the same material that we just did in Excel and in R but with a slightly different emphasis. We are going to cover the fundamentals of building regression models in Python, but we are also going to spend a bunch of time talking about an informed choice between these technologies Excel, Python, and R. We will also discuss how Python sets up the regression problem as an example of machine learning. Because we've already covered many of the intricacies of regression not once but twice, this module will be relatively short. So let's get started. Let's think about all of the possible tools in which one could implement a regression analysis ranked in order of ease of prototyping. Excel will probably be the oar on the right. It's the easiest and best prototyping tool out there. R is probably not quite as good. And old school languages such as Java or C++ are probably the oar on the left. Prototyping is not really what they are built for. In this continuum, Python probably lies somewhere in the middle. It's not quite

as good as either Excel or Right for prototyping, although, it is still obviously a lot better and more succinct than all the general purpose programming languages. And correspondingly, Python is also a lot better than Excel and R in terms of robustness and reuse. For production style environments where heavy duty modeling is going on, Python is probably the best choice. Most tools or technologies that have gained wide acceptance adhere to this principle that the common use case should be easy and the difficult use case should be possible. So let's understand what the common use case in Python is. Excel and Python both can be thought of as general purpose technologies. R, on the other hand, is more specifically geared towards statistical analysis. Now of course, even though Excel and Python are both general purpose tools, they are used in very widely deferring situations. So a typical use case for Excel might include creating a regression analysis portion of a presentation. A typical use case for R could be a case study from an academic seminar. And the corresponding use case for Python might be a trading model that scrapes websites and collects information to build models that combine both sentiment analysis and regression. This is a fair generalization of the kinds of use cases that these three tools are put through, and in particular, the reference to sentiment analysis in the context of Python is no coincidence. We've already discussed that regression can be thought of as a crossover hit from the world of machine learning and indeed Python has excellent support for machine learning and it generalizes that support from other techniques into regression. We had introduced the idea for regression as a machine learning technique by drawing an analogy with classification. Let's say we need to classify whales as either fish or mammals. One way of carrying out such a classification would be the use of human experts who would know the precise genes and sub genes that whales belong to. Another way the machine learning deals with would be to take a corpus of animals, pass in all of those animals and their attributes into a classification algorithm, and get back a machine learning-based classifier. Exactly analogously, we can create a regression model following the same process. The algorithm is now a regression algorithm rather than a classification algorithm and the corpus consists of points in the XY plane and the output of the regression algorithm are the constants A and B. This in turn, gives us a regression line with the equation y is equal to A + Bx. And we can now go ahead and use that regression line in order to work with new data. A new value of X can be passed in and the output from this will be the new predictor value of Y often by plugging the new value of X into the regression equation.

Using Pandas and NumPy

In the course of this demo, we are going to make use of no less than four important libraries available in Python. We will start with the use of pandas, which is a library that provides data

frames that very much resemble those in R. We will use numPy, which has very good support for arrays and for element-wise operations. This will come in handy while converting prices to returns. Once that's done, we make use of scikit which has regression capabilities, as well as a host of other machine learning algorithm implementations. And lastly, we make use of matplotlib, which has excellent support for 2D graphing. So let's plunge right in. Let's first start by importing pandas and numpy. Once that's done, we create variables with the names of all our csv files. These are the same csv files that we had used while working in both Excel and in R. Once that's done, we invoke pandas. We invoke the read_csv function which returns a data frame. Almost exactly like the read table command in R, this function allows us to specify a separator, as well as which columns we really care about. And just like data frames in R, it has a head method which allows us to quickly inspect just the first few elements in our data. And here we can see that just like our data frame in R, we have data arranged in rows and columns. The columns are equal in length and they all have the same type. So this bit so far marked the appearance of the first important library pandas which we use for its excellent support for data frames. Now that we have the prices nicely lined up, we've got to convert these two returns and this is where we make use of numPy. Basically, we convert individual columns of our data frame into arrays and then we divide one by the other in order to get the returns. Notice also how we use the np. array command in order to convert a column of our data frame into a numPy array. Indexing, particularly negative indexing, works a little differently in Python than in R, so let's make sure we understand what's going on. Negative indices in R are used to specify rows of columns that we wish to exclude from our data frame. And by the way, also remember that data frames are indexed starting from 1, as is common for data formats in R, whereas Python data representations are indexed starting from 0. Here, in this example, we have already converted our data frame into a numPy array, so we no longer have names for our rows and columns. But more importantly, the indices and negative indices work quite differently. There are forward indices and backward indices in Python. Forward indices start from 0 and go up to n-1. The backward indices work from the back of the data structure, so they start at -1 and go up to -n. And with that knowledge, let's take another look at the line of code in which we can work prices to returns. There's quite a bit going on in screen now, so let's break it down. On the right-hand side of the equation, we have a ratio between 2 arrays and then we subtract 1 from that ratio. Each term in the ratio, both the numerator and the denominator, consist of a numPy array obtained by converting a field from our original data frame and that field is the row containing the prices. Conceptually, what's happening is we have a formula which takes two subsets of the prices array and it then takes the ratio of those two and subtracts one from it. The first subset consists of all of the elements from 0 through n-1. This excludes the n or the last element because an index of -1 is equal then to a

positive index of n. That explains the numerator in our ratio. Let's now talk about the denominator. Here, once again, we take the prices array and we slice such that we have all but the first element. This bit is pretty straightforward. The forward index is 1. We then subtract 1 from this. Both the division and the subtraction on the right-hand side of the equal to sign happen element-wise. This has the effect of calculating percentage changes in each of the elements in our original array. And that's what gets stored in the one-dimensional array that we have on the left of this equation called returns. This array has indices from 0 through n-1. In other words, it has one less element than the original prices array. We have already seen how pandas was used and this was the start term for NumPy. It allowed us to convert prices into returns relatively easily. Let's now go ahead and pop this array of returns back into our data frame. However, because all of the elements in a data frame need to have the same number of rows, we've got to append a special value nan onto the end of our returns array. Once we do that, we can eyeball our data, and as you can see, we have all of the returns nicely lined up with all of the prices and the dates. And the last value in the returns column is the special value ne. nan. Let's go ahead and encapsulate all of the code that we just wrote into a function called readFile. This function takes in one argument, that's the filename, and it returns a nice data frame which has both the prices, as well as the returns. And then we go ahead and invoke this function with all of the data both for Google and for the NASDAQ and we store these in data structures called googData and nasdaqData respectively. We use the head command in order to eyeball these and everything looks okay. These values actually agree the value with the values from our Excel spreadsheet.

Linear Regression as Machine Learning

We are now almost ready to feed our data into the regression algorithm and get back a regression model. But before that, we need to get our X variable into the correct format for the linear regression object and that's what done on screen right now using the method reshape -1, +1. Let's understand what this is all about. Remember our discussion of how Python views the regression problem as an example of machine learning. So it takes in a corpus of three classified X and Y points and it applies a regression algorithm, probably least squares estimation, and outputs values of the constants A and B. This shows us the regression line that we can use as we'd like to. Now there is a little detail here that's worth mentioning. The sklearn linear regression object that we're going to use expects the X and the Y data in a very specific format. Specifically, it expects the X data as an array of one dimensional arrays. You might wonder why this is the case. It's because in general this is a format that is easy to specify feature vectors in. We could specify data in a similar format if we were getting a different machine learning problem, such as classification,

for example. The Y variable can be specified in whatever format we'd like, for instance, just a column of a Pandas data frame. Transforming the X variable so that they are in this rather peculiar format requires us to use the reshape -1, 1 command. This is a method which takes a one-dimensional array and gives back an array of arrays where each array inside the output has just one element. This is exactly what are going to do with our X data. We are going to feed in all of the X variables as the column from our Pandas data frame and what we get back is data in the form of an array of arrays. We don't need to jump through these hoops in order to set up our Y data. Notice, however, that while setting up both the X and the Y data, we exclude the last element from our Pandas data frame because that was a special value nan and that can lead to problems when we invoke the regression algorithm on the data. Let's take advantage of the interactive shell to eyeball what our X data looks like. And as you can see, it's an array of arrays. Each of the inner array elements consist of just one element. The Y data on the other hand is still just the column from our original Pandas data frame. Now that we have the X and the Y data set up directly, we are ready to actual invoke the linear regression model on it. And this is where the third layer on our dream team makes its appearance. We will now use Scikit for regression. We start off with a couple of imports from sklearn. We then instantiate the appropriate object and this is an object that's called linear_model. LinearRegression. We save this inside a variable and we go ahead and invoke a method on this variable. This is the fit method where we specified the X and the Y data and what's returned is a linear regression object. There are a ton of little settings which can be _____ in this call of the fit method, but here we've only specified the X and the Y data and what we've gotten back is effectively a regression line. Let's also measure the R-square. We can do so using the score method and this R-square of 0. 386 is very familiar to us. We have now come across it both in Excel and in R. The linear regression object has attributes. For instance, coef attribute returns the coefficients. This returns an array with only the value of B. This is 1. 06. Once again, that's familiar to us. This object also has attributes for the residuals and for the intercept. Those are named residuals_ and intercept_ respectively. And now that we have successfully implemented regression in Python, it's time for us to examine the residuals and some of its other properties using lock functions. So here is where Matplotlib makes its appearance. We start with an import from the matplotlib library, first, the scatter function in which we specify the X and the Y values, and then in order to also get a trend line on the same plot, we invoke the plot method on the same object. Notice how we invoke the predict method on our linear regression object with the xData specified as the parameter. We don't particularly care about X, so let's just set an ME array for both the X and the Y tick marks. And now that we've actually set up the two plots, all that's left for us to do is to invoke the show method on the plot and it appears as you can see. That was a quick run-through of the functionality for regression in Python. We made use

of Pandas because it's an excellent abstraction for the data, very much like our data frame in R. We use numPy for its support for arrays and the element-wise operations which we use while calculating returns from prices. We then made use of the linear regression algorithm from scikit. And lastly, locking functionality from matplotlib. We are now entirely comfortable using simple regression in Excel, Python, and in R. So let's now move on and talk about multiple regression, which is a lot more complex. But before we do so, let's quickly summarize all that we learned here. In this module, we understood when Python is the most appropriate tool for regression and we also went ahead and actually built a regression model in Python using some important Python libraries.

Understanding Multiple Regression Models

Introducing Multiple Regression

Hello, and welcome to this module on Understanding Multiple Regression Models. In this module, we will discuss extending regression to dealing with multiple explanatory variables. This is known as multiple regression. We will discuss how to interpret the results of a multiple regression. We will move on and discuss the risks that accompany using multiple regression. And then we get a sense of the versatility and power of multiple regression we will talk about using categorical variables and regression analysis. Okay, let's plunge in and let's start by introducing multiple regression. There is a famous idea that a butterfly flapping its wings in Brazil can cause a tornado in Texas. This is known as the butterfly effect, which is the idea that small causes can lead to large effects. This idea originates in chaos theory where it was found that running complex simulations can lead to very, very different outcomes based on just very small changes in the input arguments to the simulation and this idea is surprisingly relevant to regression analysis as well. In simple regression, we were attributing one effect to one cause. We had one independent variable, which was the cause. Changes in this independent variable drove changes in the dependent variable. And as a result, we were able to plot our data on a two-dimensional diagram, such as a plane. But in reality, life tends to be a lot more complex. Most phenomena observed in the real world have multiple causes and it will be simplistic to attribute them entirely to just one cause. And this gets us to the need for multiple regression where we have not one, but multiple causes, multiple independent variables that drive a single dependent variable. This requires us to extern that

analysis from all dimensions into three or more dimensions. This makes it a lot more difficult to visually represent multiple regression in a diagram, but most of the intuition that we have developed while discussing simple regression will easily extend to multiple regression. Let's go back to simple regression where we represented all of our points using two coordinates, X and Y. If we switched to multiple regression in three dimensions, we basically have got to add a third coordinate that's a Z coordinate to each point. In simple regression, the regression problem was to find the best fit line such that all of the points were in a sense as close to that line as possible. In multiple regression, the idea is very similar. We now want to find the regression hyperplane once again such that the distances of all of the points from this particular plane are as small as possible. Let's turn back to a little bit of linear algebra and work in simple regression. Here, we were representing each value of the dependent variable Y in terms of the corresponding value of the independent variable X. Given a set of endpoints, this gives rise to a system of equations. Now ideally, you would want to find values of A and B such that all of these equations are satisfied with perfect equality, but because we have just two free variables A and B, this is usually not possible. Therefore, we have got to live with errors. In each one of these equations, these are the terms e1, e2, and so on. Adding these into each equation leads to perfect equality. These terms e1 through en are the residuals and we want them to be as small as possible. We have to find the values of A and B which in a sense minimize the errors in our regression line. Let's take a practical example. Let's say that we want to represent the returns in a particular stock, let's say Exxon, in terms of the returns on some other financial asset, such as the Dow Jones. Now Exxon, of course, is one of the largest oil companies in the world and the Dow Jones is a large cap US equity index. This regression equation has one cause and one effect. It's basically saying that changes in the price of Exxon's stock are entirely driven or entirely explained by changes in the value of the Dow Jones equity index. But our intuition tells us that this might be incomplete. Because Exxon is a large oil company, it's also likely that its stock price is sensitive to changes in the price of oil and this is something that we can express by making use of multiple regression. We now have two explanatory variables. The first of these is the Dow and the second of these is oil. We have extended the equation that we had for simple regression by adding one additional term. This is the additional term for returns of percentage changes in the price of oil and this also gives us one additional parameter that we need to estimate, that parameter is C. Generalizing from this specific example about Exxon, for a regression, a multiple regression with two explanatory variables, we need two coefficients corresponding to those two variables, as well as the intercept A. This gives rise to a system of equations that you can see on screen now. In order to solve this, we have to estimate the values of the parameters A, B, and C. Now ideally, we would want to find values for A, B, and C such that there are no errors such that every term in Y is perfectly

expressed in terms of X and Z, but this may not happen. There will quite likely be some residual values left over that we need to add onto the right-hand side of our equation to exactly match the values of the dependent. Then the regression problem is to find the values of A, B, and C such that those residual values are minimized and this once again brings us back to the way we have solved this simple regression problem by minimizing the sum of the squares of the residuals. This approach work for two causes. We can easily extend the same analysis to any number of explanatory variables. Generalizing to key such variables, our problem becomes to estimate K+1 parameters, that is K parameters coefficients for each one of the cause entry variables and one additional parameter for the intercept. And once again, we want to select the values of these K+1 parameters such that the errors are minimized. As we had discussed in the context of simple regression, there are different methods of estimation. It turns out that the method of moments doesn't work quite right. The method of least squares, as well as maximum likelihood estimation both work just fine. And so, these cookie cutter statistical techniques are used by packages such as Excel or R in order to provide multiple regression functionality. In any case, all of this estimation is carried out for us by these packages. This is just something that we've got to be aware of.

Risks in Multiple Regression

Multiple regression is much more versatile and powerful than simple regression, but it is also accompanied by significant risks. Let's try and understand what some of those risks are. Remember again that simple regression refers to data in two dimensions which can be represented in a plane. Multiple Regression refers to data in more than two dimensions, which needs to be represented in hyperspace. The risks associated with simple regression are also relatively simple. These risks, of course, exist, but they can usually be mitigated by analyzing the R-square and the residuals of a regression. The risks associated with multiple regression are more complicated and require a little more interpretation of regression statistics. Let's quickly recap some of the risks associated with simple regression. The first is risk associated with the absence of any cause-and-effect relationship. The second is when that relationship exists, but is mis-specified. And the third is when the relationship is incomplete. As we've already seen, the absence of a cause-and-effect relationship can be detected from a low R-square or from plotting a scatter plot of X and Y. Mis-specified relationships often manifest themselves using a high R-square with residuals which are not independent of each other. And if that relationship is incomplete or missing important explanatory variables, we typically see low r-squares and residuals that are not independent of X. Mitigating these risks is relatively simple. If there is no cause-and-effect

relationship to begin with, we've just got to go back to the drawing board with new choices of X and Y. If the relationship was mis-specified, it's usually sufficient to just transform our variables by using log algorithms. And if the relationship that we specified was incomplete, then we basically want to add more X variables, ie, we have got to move from simple regression to multiple regression. All of these risks, more or less, continue to exist with multiple regression, but there is now a big new risk that we have got to understand. And that risk is multicollinearity. This is a problem which are caused when we specify multiple X variables which contain the same information. This brings me to a famous quote from General Patton. If everyone is thinking alike, then somebody isn't thinking. That can be a good thing either in life or in regression. In multiple regression, if we include multiple X variables which very closely resemble each other, then we are going to weaken the results of our regression. The regression model that we get as the output will not be very usable either for explaining variance or for making accurate predictions. That's the general idea behind multicollinearity. Let's understand the map a little better. We've already discussed how we end up with a matrix of the explanatory variables that's on the right-hand side of this equation on screen now. Multicollinearity occurs if for any two columns in our explanatory variable matrix, those two columns end up looking very much like each other. And when we say that they look very much like each other, we mean that they are highly correlated, or in terms of regression, but if we regress one or the other, we will end up with a high r-square. And remember that multicollinearity can occur with any pair of explanatory variables. This is why it's such a serious problem. As multiple regression models become more and more complicated adding three or four or five variables, you've got to check these variables in pairs to make sure that none of them are too closely correlated to any of the others. Ideally, given any pair of regressors, we would like a scatter plot that looks like the one on screen now with a low R-square and high random residuals. This is a sign of uncorrelated explanatory variables and a sign that our multiple regression is going to be robust. Let's make this a little more real using an example. Let's say that we try and add here a multiple regression to explain the returns of Exxon stock using the returns of the Dow and percentage changes in the price of oil. Here we have two explanatory variables, the returns on the DOW and percentage changes in oil. When we plot these one against the other, we find that they are mostly uncorrelated. Performing a regression of one and the other would lead to relatively low R-square. This is a good sign because it implies that our explanatory variables are different from each other, they contain different information. So this is a sign that our proposed regression model is fine. Let's say that instead of having oil prices as our second predictive variable, we've tried to explain changes in the price of Exxon using the Dow and the NASDAQ. These are both US equity indices which move very much in sync. And we should know almost instinctively that there is going to be a risk of multicollinearity here. If we plot changes in

the Dow versus changes in the NASDAQ, we get points which can be arranged in a very nice straight line. This means that these predictors are very highly correlated to each other. Regressing one on the other would give us a very high R-square. This is a tell-tale sign that multicollinearity is going to become a problem for us and we are to treat our model accordingly, more on exactly how in just a little bit.

Detecting Multicollinearity

Now that we've understood what multicollinearity is, let's very quickly understand why multicollinearity is so bad. Multicollinearity kills the usefulness of regression. It turns out that a multicollinear regression model is useful in neither of the two main use cases for regression. If you tried to explain variance using a regression model that has been built for this flaw, both the r-square, as well as the individual regression coefficients are going to be quite unreliable. We have more to see on the interpretation of the regression coefficients in just a little bit. But you should be aware that reading too much into a flawed regression model can badly misinterpret variation in your variables. The second problem which multicollinearity causes is inaccurate predictions. multicollinearity can be part of as a form of o-fitting. It leads to a model which works extremely well with the data that you already have that's known as the sample. But this mortal then ends up performing really badly when it is made to run on out-of-sample data. Think of such a model as an overpaid athlete who excels in training, but just cannot perform in a real match play. There is a lot of mathematical theory behind why multicollinearity is bad, but let's skip back and move onto the intuition of preventing and curing multicollinearity. It turns out that there are three broad classes of solutions, the common-sense solutions, the nuts and bolts solutions, and the heavy lifting solutions. In my opinion, the common sense solutions tend to be the most effective. These rely on really understanding the data on having a big picture sense of what variables are causes and effects. The nuts and bolts solutions mostly relate to setting up the data right. We'll talk about those in future sections, not right here. And then there are the heavy lifting solutions, mostly factor analysis methods, such as principal components analysis. These are in a sense a way of quantifying the common sense. Here, the basic idea is to think really deeply about each X variable, figure out which of them are closely related, and eliminate the redundant ones. This usually involves digging down to the underlying causes, factors which drive multiple X variables and which we can then include or measure directly. Let's see how you would go about applying this common sense to our Exxon regression model. Let's say that we started with this obviously flawed regression equation, which includes the DOW, the NASDAQ, as well as the price of oil. Even really rudimentary analysis will tell us that it's probably redundant to have both the Dow and

the NASDAQ in there. The Dow consists of 30 large-cap US stocks. The NASDAQ 100 consists of 100 large tech stocks. Do we really need both the Dow and the NASDAQ as explanatory variables. The answer is probably no. If we have some specific reason for keeping both, then we can construct a new explanatory variable of the difference between the Dow and the NASDAQ returns and include that instead. We really need to have a strong reason for doing so. Otherwise, including the NASDAQ in this model explaining the returns of Exxon, a large oil company, just does not make intuitive sense. This might seem obvious, but it can get very tempting to throw everything, to throw the kitchen sink into a regression because it feels like every added variable is increasing the R-square. We'll have more to see on that fallacy in just a moment. Let's now say that we have eliminated the NASDAQ, so we now have a new regression equation including only the Dow Jones, that's the returns on the DOW, and oil prices, that is percentage changes in the price of a barrel of oil. Let's now try and figure out what the underlying factors are that drive both of these factors and it turns out that a good list is GDP growth, interest rates, the level of the US dollar, and seasonality. You might ask where did I get these factors from, and this is the tricky bit, these come from a deep understanding of the data. There is really no shortcut to getting to the underlying factors. You've got to deeply understand the X and the Y variables to develop this intuition about what the underlying causes are. Now that we've identified these four underlying factors, we can go ahead and analyze each of them. We conclude that we only need to include two out of these four, namely GDP growth and interest rates. Further, we conclude that if we include these two explanatory variables, we no longer require the price of oil, but we still require the Dow Jones to be in there. All of this analysis, once again this is based on common sense and on intuition, gives us a new regression equation that you see on screen now. This is an equation where there are three predictors exactly as before, but now these are changes in the Dow, changes in interest rates, and changes in the GDP. And chances are that this regression model will perform much, much better particularly with out-of-sample data than the original regression equation that we had proposed.

Multicollinearity: Prevention and Cure

We have discussed the importance of common sense and of really knowing your data. In the second category of nuts and bolts techniques are the little details, stuff like subtracting the mean and dividing by the standard deviation in your variables. We will discuss these in more detail when we get to building regression models in Excel, Python, and R. The third set of techniques for dealing with more equally in reality includes more than one statistical technique such as factor analysis. These are in a sense a way of quantifying common sense given a set of highly correlated

X variables. Factor analysis will try and find the underlying factors that drive all of them. Principal Component Analysis, or PCA, is perhaps the most popular specific factor analysis technique out there. Let's make this a little more real using an example. Let's say that we are trying to build a regression model explaining a set of changes in home prices over time. Now we know that interest rates play a crucial role in determining whether home prices or go down and let's say that we are seeking to build a model that explains home prices only using interest rates. The problem now is that there is a plethora of interest rates out there in the market. The question that's confronting us is which of these should we use and which of these should we exclude. When we go ahead and plot all of these interest rates, we find that they are extremely highly correlated. We know right away from looking at this graph that we are going to have very serious multicollinearity problems on our hands if we attempt to run our original regression. And this is where a two-leg factor analysis comes to our rescue. We can take 20-dimensional data, we can take the average includes 1 column for every interest rate out there. We plug this into our factor analysis implementation and that will return to us say three-dimensional data. These three dimensions include three columns, one for each factor. The beauty of the magic of factor analysis is that each of these factors is guaranteed to be uncorrelated to all of the others. And what's more, if you were to regress those 20 dimensions on the left versus these 3 factors, almost 97 to 98% of the variation of those 20 interest rates would be explained using these 3 factors. The map behind this is beyond the scope of this course, but once again, we can import as many interest rates as we want and then maybe say 20 or more into the factor analysis and the output will be a matrix with 3 columns, 1 for each factor. And because factor analysis has taken 20-dimensional data in and given us equally informative 3-dimensional data out. Factor analysis is known as a dimensionality reduction technique. Dimensionality reduction is an important application in many fields, many parts of machine learning, but a simple component analysis is an excellent way of accomplishing this for regression. The output of the factor analysis is a small number of factors which represent the underlying causes in all of the data. In the case of interest rates, these factors actually have clear economic interpretations; however, this is not always the case. There is no general guarantee that the factors coming out of a factor analysis will make intuitive sense. However, with interest rates, the three key factors correspond to the level, the slope, and the twist of interest rate curves. The level is a measure of how high interest rates are, the slope measures how steep the yield curve is, and the twist refers to the convexity of the yield curve. Don't worry if these terms don't make a whole lot of sense to you. The general point is that factor analysis sometimes, but not always yields factors which have clear intuitive interpretations as well. And so, as a result of our factor analysis, we have abandoned our original regression equation which featured a whole bunch of interest rates as predictors and we have now moved to a new revised

regression equation where the three explanatory variables are the level, the slope, and the twist. Where do the values of the level, slope, and the twist come from? Well, they are the output of the factor analysis. And in this way factor analysis has accomplished the same end objective as the common sense solution. By using factor analysis with an automated procedure with a cookie cutter statistical technique, we have arrived at a very deep understanding of our data and we have significantly improved our regression model and mitigated the risks of multicollinearity.

Benefits of Multiple Regression

Let's now talk about some of the benefits of using multiple regression. These are actually quite similar to those of using simple regression with some added nuances. We have discussed how simple regression is great because it's powerful, versatile, and deep. Powerful because it's perfectly sorted to two common use cases. Versatile because it can easily be extended to non-linear data and deep because it is an adaptive item that changes based on the data that it is operating on. Each of these attributes is also present with multiple regression. Multiple regression is even more powerful because it controls for the effects of those different causes. It is significantly more versatile than simple regression because we can easily extend multiple regression to work with categorical data. This is where our explanatory variables take only finite values, such as days of a week or months of a year. And lastly, multiple regression is also very deep, particularly when combined with more advanced techniques such factor analysis. We have already discussed the part of these points in the context of multicollinearity and we'll talk about the second of these points at length while discussing categorical data and dummy variables. For now, let's focus on the first. Regression, whether it is simple regression or multiple regression, tends to be used mostly either to explain variance or to make predictions. Now multiple regression has one great advantage over simple regression. Multiple regression allows us to make controlled experiments. It allows us to send how much our dependent variable will change for a unit change in one out of many predictor variables keeping all of the other variables constant. Let's say that, once again, we are explaining returns in Exxon stock using the Dow Jones and the price of oil. Multiple regression will allow us to answer questions like all else being equal, how much will Exxon stock move by if oil prices increase by 1%? Notice that this is a subtle and a deeper question than asking how much of the variation in Exxon stock is explained by oil and by the Dow. Let's do a little bit of simple arithmetic. Let's plug in the new values for the Dow and for oil into our regression equation and find the corresponding value of Exxon stock. We have a question mark as the sub script denoting this because we don't really know or care when in the future this will occur. We can keep the value of the Dow as exactly the same as in the original

equation and only increase the value of oil by 1%. Then the change in Exxon stock can be calculated by subtracting one equation from the other. All of the terms will cancel each other out with the exception of the value C multiplied by 1%. And because the units of the oil term were already in percent, really the change in Exxon is simply the constant C. There is an important insight here. The regression coefficient for any explanatory variable tells us how much Y will change for a unit change in that explanatory variable, all of the other X variables being held constant. And this ability to make control comparisons is an important part of the appeal and the power of multiple regression.

Interpreting the Results of a Regression Analysis

Let's now talk a little bit about interpreting the results of a multiple regression analysis. It turns out that this is significantly more complex than interpreting the results of a simple regression. In the case of a simple regression, really there are two factors that we've got to pay attention to, the R-square and the residuals. The R-square measures the overall quality of fit. The higher the R-square, the better. There is a little asterisk here that if we get a very high value of R-square that is about in 90%, then we've got to satisfy ourselves that we have set up the regression correctly. We might find that the residuals are not uncorrelated. In any case, the combination of the R-square and the residuals together are enough in the case of a simple regression to tell us if there is anything seriously amiss with our regression. There are additional statistics available, such as the standard errors of the individual coefficients, but in reality, these are rarely used. The situation is very different and a lot more complicated when it comes to interpreting a multiple regression. There is a whole host of statistics that we've got to know how to interpret starting with the adjusted R-square. This is a variant of the R-square, but with a penalty applied that penalizes a regression model that has included irrelevant variables. Next up, there are the residuals which continue to be very important also in multiple regression. Then there is something known as the F-statistic. This is a very powerful statistic, but it requires a fair bit of statistical knowledge to interpret this correctly. In Excel, for instance, there is no direct interpretation available. We'll talk about this in greater detail while discussing the R in the Excel versions of multiple regression. And there are also the standard errors of the individual parameters of the individual coefficients. These tend to be ignored while studying or analyzing simple regressions, but they are reasonably important while analyzing a multiple regression. There is a simple rule of thumb that we can use here. The value of any coefficient divided by standard error should ideally be greater than three. If that ratio of the coefficient divided by standard error slips below one, then we should almost certainly omit that variable from our regression. And then the last metric is the plain word R-

square. This is not a reliable metric in the case of multiple regression. Unfortunately, Excel insists on reporting only the R-square and not the adjusted R-square and this is quite a problem. In any case, the R-square is misleading with multiple regressions because it only goes up as we add more and more variables; however, as we add more and more irrelevant variables, we are reducing the quality of our model and increasing the risks of multicollinearity. So a straight reliance on R-square, in the case of multiple regression, is quite dangerous. Let's dig just a little deeper into R-square and adjusted R-square. Let's start with a refresher of our introduction of the concept of R-square. Remember that the residuals of any regression are the difference between the actual and the fitted values. By calculating the variances of both the left and the right-hand sides of these equations, we get a relationship of decomposition of the variance of our dependent variable into the variance of the fitted values and of the residuals. There is a little bit of statistical stuff going on behind the scenes which simplifies this equation. That's because the covariance of the fitted values and the residuals that are always 0. Never mind if you don't quite know what that means. In any case, the result is that we can decompose the variance of our dependent variable and pull the variance of the fitted values and the variance of the residuals. There are specific names for each of these three terms. If we then go ahead and take a ratio of variance of the fitted values or the variance of the original dependent variable, that ratio is called the R-square. The higher that ratio is, the better the quality of the regression. The R-square is a number that always lies between 0 and 100%. The higher the R-square, the better. This is for simple regression. However, for multiple regression, sometimes a higher R-square is not better. That's because with multiple regression as we keep adding explanatory variables, the R-Square keeps increasing. This is the case even if the new explanatory variable that we have added is an irrelevant one. Doing so will increase the risk of multicollinearity, which as we've seen is quite dangerous and that's why we need to find an adjusted R-square which penalizes this metric for adding irrelevant variables and that is exactly what the adjusted R-square is. We will not go into the details of how we decide whether a particular variable or a set of variables is irrelevant, but I will just very quickly mention that is a criteria that depends on the F-ratio or the F-statistic and this is how the adjusted-R-square ties together the R-square, the F-statistic, and even the standard errors of the individual parameter coefficients. And for this reason, the adjusted R-square is the one number, the one metric which I personally rely on while interpreting multiple regressions. There is a small downside to the adjusted R-square, in rare instances, it can become negative, but in general, it's an awesome way to make sure that your regression model is well-specified.

Categorical Variables

Let's study how multiple regression can be used with categorical variables. These are variables which take one of a finite set of values, for instance, days of a week, gender, and so on. Categorical variables are discreet, that is they only take one of a finite set of values and like all of the X and Y variables which we've been discussing so far which have been continuous. This can be a little abstract to understand, so let's make a trail starting with an example. Let's say that we are trying to measure what drives the height of individuals, and so we propose a simple regression model like the one you see on screen now. We have a simple predictor variable and that is the average height of that individual's parents. At first glance, this might seem like a perfectly reasonable regression model, but when we plot the values of Y versus X, we find a rather surprising pattern. We see that the data arranges itself nicely into two linear portions as you see on screen now. Effectively, there are two straight lines here, one for males and the other for females. Now if we were to solve the regression problem and find the best fit line, it actually would not have a great R-square because the regression line is now quite far from all of the points in our data. This regression line, by the way, has the intercept A. Really what we need is one line for males, which passes through all of the points corresponding to the males and has an intercept A1, and another line for the females, which has the same slope, but has a different intercept, A2. As we can see, the two lines for the males and the females have the same slopes, but they have different intercepts, A1 and A2. And this is a perfect use case for the use of something known as a dummy variable. Using a dummy variable, we can combine these two equations into a single combined regression line. This is what you see on screen now. The dummy variable here is the term D. A dummy variable is a special kind of categorical variable which takes only 1 or 0. Here, the dummy variable D is 0 for males and 1 for females. Plugging this in, we can see that our combined equation works out correct for both males and females. If a particular individual is a male, then the value of D will be 0 and the combined regression line will work out to A1+Bx, which is exactly the line that we had for males. Similarly, if an individual is a female, D will be equal to 1, and as a result, the combined regression line will evaluate to A2+Bx, which again is exactly the equation for the females. And so, this clever little trick of introducing a dummy variable which takes either 0 or 1 as its value has allowed us to form a combined regression line which works for both of these groups in the data. One important observation here, our data contained two groups, and so we needed to add one dummy variable to our equation. Generalizing this, if we have data with k groups, we need to set up k-1 dummy variables. This is an important point. If we go ahead and add k dummy variables, instead of k-1, this is a common mistake, we end up with multicollinearity. Having one less dummy variable than the number of groups we have makes sure that there is no multicollinearity between the intercept term and all of our dummy variables combined. Generalizing this idea, we can use it to represent all kinds of

categorical variables in a regression. Dummy variables are a special case of categorical variables. They only take the values 0 or 1. In that sense, they are binary variables. Categorical variables are more general, but they all only take a set of finite values. For instance, a categorical variable might represent the day of the week, the month of the year, or even just a binary data item such as gender. To include non-binary categorical variables, we simply need to set up the appropriate number of dummy variables. Let's say, for instance, that we want to represent the quarter of the year in a regression with the stock returns. We might set up a regression equation in this form if we are investigating window dressing by mutual funds, for instance. This is the phenomenon where financial institutions tend to buy up stock towards the end of the year in order to win the rest their returns. Here, the data has four groups and so we add three dummy variables, Q1, Q2, and Q3. Each of these dummy variables is 1 only if the date is in the corresponding quarter, and it's 0 otherwise. For instance, Q1 is 1 only for the months of January, February, and March. This is a simple way to extend multiple regression to work with arbitrary categorical data. We can also use multiple regression when the different groups have different slopes. If, for instance, we found that the equations of the height lines for males and females had not only different intercepts, but also different slopes, we could extend that clever trick and set up a combined regression in the form that you see on screen now. Here, we have dummy variables, not just for the intercept, but also for the slope. In particular, the dummy variable D2 takes the value 0 for males and it takes the value X for females. The beauty of this combined regression line is that if we plug in the appropriate values of the two dummy variables for males and females, we will get back the original correct regression lines for the two groups. A1+B1x for males and A2+B2x for females. Let's take a second to see how the equation for females works out just right. We plug in the value of B1 equal to 1, and of D2 equal to x. And once we go ahead and rearrange the terms, we are left with the correct equation, A2+B2x. These examples demonstrate the flexibility and versatility of multiple regression. We have seen how we can use dummy variables where the predictor variables or the X variables are dummies. We have not discussed a different case where the dependent variable or the Y variable is a categorical variable. That is a class of techniques known as logistic regression and that's beyond the scope of this course. We are now going to move on and apply build multiple regression models in Python, in R, and in Excel. But before that, let's summarize all that we just learned. We began by understanding the formidable benefits of multiple regression and also some of the significant risks that come with those benefits. We understood the utility of the adjusted R-square measure. And we rounded out this conversation by demonstrating how categorical variables can be included in the multiple regression model.

Implementing Multiple Regression Models in Excel

Introducing Linest in Excel

We've gotten a good grasp of the theory of multiple regression, this built upon a knowledge of both the theory and the implementation of simple regression. In this module, we are going to move onto multiple regression and show how to implement it in Excel using the LINEST function which isn't the most user-friendly function out there. We will also learn how to interpret the results of a multiple regression. This is significantly more complex than interpreting the results of a simple regression. And we wind up the module by discussing the use of categorical variables in regression analysis and demonstrate how this can be carried out in Excel. Let's get started with a quick look at how we implement multiple regression in Excel. We've already worked with a bunch of regression-related functions, slope, intercept, Rsq, which returns the R-square, and forecast, which we can use for prediction. But really, the workhorse functions for multiple regression in Excel are linest and logest. Linest, as its name would suggest, is a linear estimator. Logest is a logarithmic estimator. Linest is basically a linear version of logest. We will discuss linest in a great deal of detail. Virtually, everything that we see about linest will also be applicable to logest with the only difference being in the functional form of the curve that they fit. Let's say that we'd like to use multiple regression in order to explain the stock returns of Exxon. We feel that these stock returns can be explained using the returns on the S&P 500, which is an important global equity index, and also by changes in the price of oil. It can be a little complicated to get accurate historical prices for oil. We are going to use a large oil ETF for Exchange Trader Fund. You can think of this as a stock-like instrument which is meant to track the price of oil. And we can use the linest function in order to carry out this multiple regression. And so, we are going to specify a multiple regression model with one Y variable returns in Exxon stock, and two X variables returns on the S&P 500 and on the USO ETF. In our example, for the value of the known_y's, we will select in Excel the column or the row containing the returns of Exxon. The next input into the linest function are the known_x's. In this example, we are going to specify two columns. These contain the returns for the S&P 500 and the USO. The known_x's column is option. If this is omitted, the values one, two, three, and so on will be used implicitly in a column of the same length as the known_y's. The third input into the linest function is a flag telling linest whether we would like a constant or not. If we specify this to be true or if we omit the value, the regression line that will be evaluated will include a constant A. You might wonder whether there are

situations when we actually want to perform a regression without an intercept and the answer is yes. In fact, we shall see an example of this while discussing categorical variables in just a little bit. Also, the R-square reported a regression without an intercept might defer depending on whether that regression was carried out in Excel or in Python. And the last input into the linest function is a flag telling linest whether we would like detailed regression statistics or not. We are going to specify this to be true. If we specify it to be false, only the regression coefficients will be reported. However, because we specify it to be true, we get back a wealth of regression-related statistics. Here on screen now is everything that linest returns. There's a lot going on, so let's cycle through it really quickly. The first rule of the linest result contains all of the regression coefficients and over on the extreme right is the value of the intercept A. After that, come the coefficients of the X variables, but annoyingly, these are specified in the reverse order from what order we had passed into the input. For instance, in this example, we had specified the values of the S&P first and of USO second; however, the coefficients reported will be USO first and the S&P second. This is actually quite annoying and it makes linest results hard to interpret. The second rule of the result of linest includes the standard errors for all of the coefficients on the first row. Think of these numbers as a measure of how uncertain we are about those coefficients. We like the standard letters to be as small as possible. On the next row, first up is the R-square. Annoyingly, linest reports the R-square and not the adjusted R-square followed by a quantity known as the standard error of the regression. This is closely related to the variance of the residuals of the regression, but it also has a special role to play in determining whether the regression as a whole has been useful or not. And that is also the rule of the F-statistic, which is quite an important statistic. We'll discuss it in detail. Next up, Excel reports the degrees of freedom. These can be calculated using the formula n-k-1 where n is the number of data points in our regression and k is the number of regressors. We will only care about the degrees of freedom in order to interpret the F-statistic. And in the last row of the regression result are two quantities we are familiar with. The first is the explain sum of squares, which is the variance of the fitted values followed by the residual sum of squares, which is the variance of the residuals. Linest is an extremely powerful function, but for a number of reasons it's hard to use. To begin with, it is an array formula, which means that we've got to select a range of cells in Excel and then hit Ctrl+Shift+Enter after typing in the formula. This makes it different from the vast majority of Excel formula which reside in a single cell rather than in an array. The fact that it's an array formula makes it awkward to use. In addition, Linest has some relatively minor issues in its regression statistics. it reports the R-square rather than the adjusted-R-square. It does not interpret the F-statistic. It's inconvenient that the regression coefficients are reported in the reverse order. And lastly, linest does not handle

missing values gracefully. Despite all of these drawbacks, linest is still a very powerful function and it's an awesome way to get our regression model up and running quickly in Excel.

Standard Errors

We've learned how to use linest and we've also seen that linest returns a wealth of statistical information. In order to actually interpret all of these statistics, we're going to need to dig just a little bit deeper into the mathematical side of things. If we look at everything that's returned from linest, we can see that we've already discussed the R-square, the explain variance, and the residual variance. That still leaves a lot for us to understand, so let's cycle through the result of linest one by one starting with the standard errors of the coefficients. The second row of the result of linest includes the standard errors for all of the coefficients on the first row. Now in order to understand what those standard errors are, we are going to have to understand the difference, the fundamental difference, between the population and the sample. You can think of the population as all data points out there in the universe and you can think of the sample as a subset of all of those data points in the population. Typically, the way we use samples is to study a sample, that is a small subset of the population, and then generalize the population at large. This is done using hypothesis testing. In hypothesis testing, it's really important that our samples be representative of the population as a whole, ie. that they be unbiased. If we draw a biased sample, which in some way does not represent the population as a whole, any conclusions or inferences that we make about the population on the basis of that biased sample will be quite dangerous. Now it's important for us to realize that regression is actually an operation that we carry out on samples and not on the population as a whole. We are given a small number of points that we can observe, and from these points we apply the least squares estimation method and come up with the best fit regression line. Let's say A+Bx. But the points that we can see really are only the sample. There are a lot more points out there. There is an entire population of points that we cannot see or do not see while carrying out our regression. Let's say that somehow we did know about these grey points and we included all of them in our regression estimation, you would end up with a regression best fit line with a different equation. That equation will be something like Y is equal to A+Bx. This is the population regression line and it follows the convention of using greek alphabets for population parameters. and English alphabets for sample parameters. Even within sampling, depending on what sample we draw, we are going to end up with different values of the constants A and B. Let's say we drew one subset of points, call that sample one, the regression equation we would get would be Y is equal to A1+B1x. If you draw a different sample, call that sample two, the corresponding regression equation would be Y is equal to A2+B2x. Now

let's say that we drew a really large number of samples and we created a histogram of the values of either A or B and this is a normal distribution. Specifically, this is known as the sampling distribution of whatever parameter we are estimating. Remember again that the sampling distribution is a probability distribution of the population parameter. So for instance, let's say the population parameter alpha is going to be represented by a sampling distribution where it's mean is the sample parameter A. In other words, we will never precisely know the values of the population parameters alpha and beta, but we can guess using the sampling distribution. And in fact, we do not even need to draw a large number of samples. We can extrapolate from just one sample and that is possible because of some really deep results from the field of statistics, results that we will completely ignore for now. Now that we've understood the idea of a sampling distribution, let's put it to use. The reason we care is that it contains within it indications as to whether our regression has been performed well or not. Our conversation here is about simple regression, but in reality, all of this can be extended to multiple regressions as well. The sample parameters A and B that we got from the least squares estimation are our best estimates of the population parameters alpha and beta. The sampling distributions of alpha and beta are going to be centered around these values A and B. These sampling distributions are normal and we've discussed the mean of these sampling distributions, but as we recall, a normal distribution is actually defined by its mean and its standard deviation. So the question that we've got to answer is what is the standard deviation of these sampling distributions. And that brings us to the idea of the standard error of a regression parameter. The standard error is simply the standard deviation of the sampling distribution of a parameter. There is some mathematical fine print that I'm going to dispense with here, but the basic idea is that given any regression line, Y is equal to A+Bx, we can estimate the standard error, that is the standard deviation, of the sampling distribution, and this standard error is directly proportional to the variance of our residuals. We will not go into the exact formula of these standard errors, but you should understand that they are linked to the size of the residuals. If we have a great regression fit with a high R-square, the residuals are going to be small, and the standard errors of all of our regression coefficients are going to be small as well. On the other hand, if we have a bad regression fit, then the residuals of this regression are going to be large in magnitude, and the corresponding standard errors are going to be large as well. So in general, the smaller the residuals, the smaller the standard errors, and the better the quality if our regression. And this reflects on the weight of the sampling distribution. If you have a low standard error, we have a narrow sampling distribution and a high level of confidence that we have estimated that parameter coefficient correctly. If you have a high standard error, you have a low level of confidence that our parameter value is correct. Y is equal to A+Bx. A and B are sample estimates which have been arrived at by solving the regression problem using an

estimator, such as the method of least squares. Ideally, we would like to find the values of constants A and B such that all of the equations above are satisfied perfectly, but in reality, we will have to live with some differences, some deltas, and these are the residuals of the regression. So we can easily calculate the variance of this set of residuals and this is known as the residual variance or RSS. From this residual variance, we can estimate the standard errors of our regression parameters. We will not go into the exact formula because these aren't important. These are cookie cutter formula that are implemented by Excel and R and so on, but the intuition is important. We can estimate the standard errors of our parameters from the residuals of the regression and the greater the variance of those residuals, the greater the standard errors around that estimates. The smaller the magnitude of our residuals, the smaller the standard errors and the better the quality of our regression.

T-statistics and P-values

How statistically significant is a given variable in a regression equation? This is a question that often comes up. In order to answer this question, we've got to use the concept of t-statistics. Linest does not directly report the t-statistics of our regression. However, we can easily calculate them by dividing a regression coefficient by its corresponding standard error. That's all there is to it. Let's understand what t-statistics are all about. Now that we have a measure of what the mean and the standard deviation of the sampling distribution, we can also start to carry out some calculations about the probability of occurrences. For instance, we had discussed that in a normal distribution, about 68% of the points were live within 1 standard deviation of the mean. And given the symmetry of a normal distribution, we can infer from this that around 16% of the points will lie to the right of one standard deviation. Notice how we have been able to make a statement about the probability of the occurrence of the position of a point given the mean and the standard deviation. And this result is used cleverly in order to test whether our regression estimation has been successful or not. We have the sampling distributions which are centered around the values A and B and which have the standard deviation given by the standard error. Now remember, we are only really interested in the standard errors because they help to tell us whether we have done a good job in estimating the value of this particular parameter. And the way in which they do so is via a quantity known as the t-statistic. The t-statistic is helping us evaluate a hypothesis. This is known as the null hypothesis, and in this instance, the null hypothesis is that the corresponding parameter was actually 0. Let's say that the actual value of alpha was 0, what then is the probability that our regression fit would have given us the value A. This is a rather abstract question and it takes some getting used to. Again, let's say that our null hypothesis was that the

value of the population parameter alpha was 0. What then is the likelihood that our calculated sample value would be A? Why 0 you might ask? Well, just because it's a handy way of measuring whether a particular parameter, a particular coefficient is adding value to our regression equation. If the actual parameter coefficient was 0, then we could just exclude it from our regression line. Now clearly, the farther away A lies from 0 as measured by standard deviations, the more unlikely that this null hypothesis is true. The best case for us is that it is basically impossible that we have estimated our parameter as E, when in reality alpha is equal to 0. And so, the farther away, the greater the distance between our parameter estimate and 0, the better our estimate is and that distance measured in units of the standard deviation is known as the t-statistic. A big t-statistic allows us to reject this null hypothesis and it's a good thing. Let's make this real with an example. Our value of the parameter A lays 0. 85 standard errors away from 0. Our estimate of the value B lays 9 standard errors away from 0 and that's why we see that the t-statistic of the parameter A is 0. 85 and the t-statistic of B is 9. 01. This helps us to answer the question, is an individual estimate of either A or B adding value to our regression. If the t-statistic is high, then the answer is yes. The higher the t-statistic of a coefficient, the higher our confidence that we have gotten the value of that coefficient right. So a t-statistic is telling us how well our regression model has performed in calculating the value of a particular parameter. Now the t-statistic is also closely linked to something known as the p-value. We go back to our null hypothesis that the original population parameter is 0. Well then what is the probability that we have observed that parameter constant to be A and this probability is the p-value. Clearly, greater the t-stat was smaller, the p-value is going to be. We want this p-value to be as low as possible and we want the corresponding t-stat to be as high as possible. The lower the p-value of a coefficient, the higher our confidence in our estimate of that coefficient.

The F-statistic

Now that we've understood what t-statistics, standard errors, and p-values are all about, let's turn our attention to the one important remaining measure of regression and that is the F-statistic. The F-statistic is a measure that tells us how good overall as a whole our regression model is. And while discussing the F-statistic, we will also understand the two remaining reports from the linest function that we haven't discussed so far, the standard error of the regression, and the number of degrees of freedom. For us to understand this, we need to clearly understand the difference between residuals and errors. Remember that we have a regression equation corresponding to our sample. This equation is given by y is equal to A+Bx. Ideally, we would like to have values of A and B such that all of these equations are perfectly satisfied or because it's unlikely we would be

able to do so, we need to live with some deviations. Those are called residuals. These are the values e1 through en. So the residuals are very much a property of the regression. There are sample properties which can be measured, and the variance of these residuals is known as the RSS. But in reality, there is also a population's regression line somewhere out there that has the equation y is equal to alpha plus beta x. Now what we'd really like to do is estimate the values of alpha and beta, but because we can only work with a sample, the best we can do is estimate the values of A and B and this populations regression line is also going to have some error terms. These are not the residuals. These, in fact, are the errors. And like all population properties, we will label them using the corresponding Greek alphabet epsilon 1 through epsilon n. Now remember that we had mentioned that we can calculate the standard errors of our regression parameters and that these standard errors are related to the variance of our residuals. Well, I was actually being just a little imprecise there. Those standard errors are linked to the variance of our errors. But because we cannot directly calculate the errors, we've got to make do with the residuals instead. But we still can estimate the error variance and this will tell us whether our regression model as a whole is a good one or not. And this brings us to a quantity known as the standard error of the regression. The variance of our errors is basically best estimated using the variable of our residuals divided by n-2 and this standard error of the regression is enough for us to calculate the standard errors of alpha and beta and of all of the regression parameters. And this standard error of the regression is quite an important quantity. It's important enough to actually be included in the result of the linest function. That still leaves us curious about whether we have actually minimized the variance of our overall regression and this brings us to another interesting result. The ratio of the variance of the residuals to the variance of the errors is distributed as a chi-square random variable. Chi-square random variables and a particular type of random variable just like normal. The curve that they follow looks quite different from a normal bell curve and that's what you see on screen now. We can use this distribution to answer the question, does our regression as a whole add value at all. Just as a t-statistic is a measure of how reliable one particular parameter coefficient is, the f-statistic is a measure of how reliable all of those parameters are as a whole. The higher the f-statistic, the more value that our regression model is adding. Remember how a t-statistic of an individual regression parameter coefficient gave us a way to reject a null hypothesis that particular parameter value was 0 exactly in the same way the f-statistic is giving us a test statistic allowing us to hopefully reject the null hypothesis that all of the regression coefficients are 0. Why 0 you might ask? Well for the same reason as before. A set of regression coefficients, all of which are equal to 0 basically have no predictive power. This is the worst case. This is the worst that we could possibly do. If we are unable to reject even this weak null hypothesis, then clearly, we are not adding a whole lot of value with the regression line.

What's the significance of that point where both data and alpha are equal to 0? Well remember again that the regression line is based on a sample and not the population. A and B are the values that we've obtained by fitting a line through the sample. These are just our best estimates of the corresponding population parameters, alpha and beta. And the worst that we could do is to simply pick a point where both alpha and beta are 0. Now if all of the regression coefficients was 0, then the total variance of our residuals would be equal to the variance of the original Y values, that is the highest possible value. However, because we actually have a much lower value of our residual variance, hopefully, we are able to reject the null hypothesis. And the further away from that peak we are, the less likely that null hypothesis is. And this gets us to the last metric that is returned by linest that we have yet to discuss, which is the number of degrees of freedom. The f-statistic just like the t-statistics has an associated P-value. This is a likelihood that the null hypothesis is true. In order to find this likelihood, we use the number of degrees of freedom. We plug these into an Excel formula called fdist. It's likely inconvenient that Excel does not interpret the f-statistic for us, but even so, we can do this ourselves if we really want to. So just as p-values and t-statistics tells us whether individual parameter coefficients are in a sense good, the F-statistic tells us whether an entire regression line as a whole is good or not.

Multiple Regression with Linest

Now that we've understood theory behind some of the regression metrics that are commonly reported, let's go ahead and implement multiple regression in Excel. And so we make use of the same csv files with the raw data for Google, Exxon, and other financial assets that we have obtained from Yahoo Finance. In this example, I've already set up data with the prices of Exxon, which is a big oil company, and the S&P 500. We've already discussed the importance of regressing returns and not prices, so I'm going to convert these prices into returns, that's the very first step before we do anything at all that's useful with it. Let's start first with a simple regression of Exxon returns and S&P 500 returns. Let's first use the slope and the intercept functions, find the regression parameters and the r-square to calculate the quality of our fit. We get an R-square of 29% and the value of the slope is 0. 594. Let's now go ahead and perform the same regression, but now instead of using the slope and the intercept functions, we will make use of Excel's linest function. Linest is an array function, which means that you've got to select an array arrange on screen, type out the formula, then hit Ctrl+Shift+Enter. This will cause all of the cells in your array to be populated with the results. Using an array function can be somewhat confusing, so let's make sure we get this right. Select the region on your worksheet which is big enough to contain the results and enter these four inputs. The first input is the known_y's. These are Exxon returns.

The second are the known_x's. Here are the S&P returns. Third is a constant which we set to one or true because we do want to include an intercept in the regression equation. And fourth is a flag also set to true telling excel that we do want detailed regression statistics. Then go ahead and hit Ctrl+Shift+Enter. Be sure that the region of the worksheet that you selected was large enough to hold the results. It doesn't matter if it's larger than it's strictly required. All that will happen is that Excel will fill in the extra cells with hash and As. But make sure it's not too small, else only the reason that you have selected will actually display. So that's what I've gone ahead and done and then I've hit Ctrl+Shift+Enter and the formula results show up. The first row includes the coefficients, the second row has the standard errors, and the first column of the third row is the R-square. We can make use of the coefficients and the standard errors to calculate the t-statistics. Remember that we simply divide one by the other. The standard error is merely the standard deviation of the sampling distribution and the t-statistic tells us how far away, how many standard deviations away from 0 the value of our coefficient lays. We go ahead and calculate our two T-statistics for the slope and for the intercept. Our T-statistic of less than one is not really satisfactory, by the way. Notice also how the values of the slope, the intercept and the R-square perfectly match the results from the original functions for slope, intercept, and Rsq that we had used while discussing simple regression. Below that is the F-statistic. Unfortunately for us, linest will not actually interpret the F-statistic, it will merely report it. Next to that is the number of degrees of freedom, that's 1. 29 and that's required for us to actually interpret how statistically significant that F-statistic is. And then the last row of the results contains the explain variance and the residual variance respectively. Let's now go ahead and add one more explanatory variable. We are going to add the returns of USO which is an important American ETF which is meant to track the prices of oil. So we are going to add two columns, one for the prices and the other for the returns on the USO asset. Then I'm simply going to copy over the prices from that other tab in the worksheet titled USO. Notice also the #N/As at the start of the USO price series and that's because USO actually started reporting a few months later than the first point in our data, so we are missing the prices of the USO ETF for the first three months of 2006. This is our first multiple regression. We are going to specify two X variables, one Y variable, and we go ahead and use the linest formula. Linest takes in four parameters, the known_y values, the known_x values, whether we'd like an intercept or not, and lastly whether we'd like to report the statistics. I'm going to set both of those last parameters to be true. Then hit Ctrl+Shift+Enter and there are the results on screen now. The first bit worth noting is that linest reports the coefficients in a rather strange order. It reports the intercept on the right, then coefficients for the corresponding X variables, but in the reverse order in which they were specified. So in this example, because our data had the S&P first and then USO, the coefficients are going to be in the reverse order. We will first have

USO and then the S&P. This ordering makes it really hard to remember or to understand how linest works and it's just a poor bit of design. In any case, the R-square of this regression has gone up as expected to 0. 32; however, that is not a very improvement if you consider that we've added another X variable. As we shall see later on, the adjusted R-square of this regression is not as high as the r-square.

Categorical Variables in Excel

Let's now move on and demonstrate how we can carry out multiple regression in Excel using categorical variables. We had discussed an example of a simple proposed regression where the Y variable was the height of an individual and the X variable was the average height of that individual's parents. We had noted that in an example like this, if we plot Y against X, we get two nice sets of linear data for male and female individuals respectively. Really here, we need not one, but two regression lines which have the same slope, but different intercepts. The regression line for males is given by the equation y is equal to A1+Bx. The corresponding line for females is y is equal to A2+Bx. And we had discussed a clever little trick which allows us to combine both of these equations into one combined regression equation. That was through the use of something known as a dummy variable D. This dummy variable D will take the value of 0 for males and 1 for females. The coefficient of this dummy variable will be A2-A1 and the regression problem now will be to determine the constants A1, A2, and D. For males, this line would work out to give A1+Bx, which is the correct regression line for males. And similarly for females, the corresponding regression line would work out to y is equal to A2+Bx, and in this way, this combined regression line will perfectly specify our data. Here our data contained two groups, and so we added one dummy variable. We had noted that given data with k groups, we need to set up k-1 dummy variables, otherwise, multicollinearity would result. Now it turns out that when we actually build regression models with categorical variables, we can also tweak this procedure just a little bit to add-in two dummy variables, but lose the intercept. In practice, this is a convention that I prefer. Mathematically, the two are equivalent. Let's see how. Let's say that we set up a combined regression line which had the equation. Y is equal to A1D1 + A2D2 + Bx. The regression problem is still unchanged. We still need to solve and find the values of A1, A2, and D. And once again, this combined regression line evaluates to the correct lines for both males and females. The difference now here is that we do not have a free intercept. Here, A1 and A2 are the coefficients of the two dummy variables. And this demonstrates how we can also choose to use k dummy variables for k groups, but we have got to be careful to exclude a free intercept. We can go with either one of these two approaches. I favor the second approach of having k dummy variables and excluding

an intercept, even though this sometimes can lead to a negative R-square. I prefer this method because I find it intuitively easier to understand the coefficients for each of the k dummy variables. Let's see how this is done in Excel. Let's say that we want to test whether Google stock returns tend to be particularly positive in any particular month of the year. You should know that this is the kind of analysis that stock fundamental analyst do. For instance, there is something known as the January effect. It is hypothesized that stock prices go up in January because of buying from mutual funds. In any case, we set up a bunch of variables corresponding to each month of the year and then we have a simple formula. This formula will set the value of a particular variable to be 1 if and only if it corresponds to that date in the return series. So for instance, for January 2006, only January will be 1. All of the other 11 dummies will be 0. We copy/paste this formula and in this way we populate our 12 additional columns of explanatory variables. Each of these is a dummy because each value here can only be 0 or 1. We then add in the NASDAQ returns because we still want to control for overall stock performance and remember how multiple regression helps to control across different variables. We then go ahead and use linest. This time, we are careful to specify that we do not want a constant. That is why the third parameter that we specified to linest is 0. All of the other parameters take their usual values. We have an R-square of 0. 49. In this example, it's also really important for us to calculate all of the t-Statistics because we will only know from the t-statistics which of our variables is significant. So we calculate the t-statistic as the ratio of a coefficient with a standard error, that's the usual formula, and we are left with 13 t-statistics in all. Now remember again that linest reports the parameters in reverse order so the parameter on the extreme right is for the NASDAQ returns. That is the very high positive t-statistic of 8. 93. That tells us that particular X variable for the NASDAQ returns is definitely adding value in our model. The only other dummy variable which has a large magnitude on its t-statistic is that for October, which has a t-statistic of 3. 65. The only month in which the month effect is statistically significant for Google's stock outperformance over the NASDAQ is October and there are some sound reasons for this. One of them has to do with the fact that many stock market crashes have tended to occur in the month of October and Google tends to be something of a safe haven at such times. And in this way, we've used linest to carry out multiple regression with categorical variables. Let's now go ahead and repeat this in both R and in Python. Excel is an amazing prototyping tool, but R really sits at the sweet spot in terms of ease of prototyping and of robustness and possible reuse. If you are going to be performing regressions in excel, you seriously should consider using R instead. R for regression makes sense no matter what your use case and we will see why in the module that's coming up next. But before that, let's summarize all that we just learned. Here, we learned how to

implement multiple regression in Excel. We interpreted the results of a multiple regression. And we extended that analysis to take into account categorical variables.

Implementing Multiple Regression Models in R

Multiple Regression in R

Hello, and welcome to this module on implementing multiple regression models in R. This is going to be a short and easy module and that's because multiple regression in R basically just works. It's effortless almost for us to interpret the results of a multiple regression in R. That makes it quite different from linest in Excel. And we will also talk about easy it is to carry out categorical analysis in R. Let's get into a multiple regression demo. Let's pick up right from where we had left off while discussing simple regression in R. Then we had created a simple R function called pre-process which takes in the names of two files corresponding to X and Y variables and this returns a data frame which has returns for both of the X and the Y variables in there. We had discussed in quite some detail what data frames are and how we use R's functionality to convert prices into returns, so I'm not going to repeat all of that. Let's carry on. Let's immediately put this function, pre-process, to use. Let's start with the same simple regression that we just attempted in Excel. Let's regress Exxon on the S&P 500. So we read in the files with data for the S&P 500 and Exxon and then we regress one on the other using R's lm method. And at the end of all of that, we invoke the summary function on the linear model object which is returned by lm. The results of the simple regression are on screen now and also we have a lot more knowledge than we did in the last module in R. We can interpret most of what's on screen There is a multiple R-square of 0. 29. There is an adjusted R-square. Notice that it's a little bit lower of. 0286. There is an F-statistic. Notice that R unlike excel also interprets this F-statistic for us because it gives us the p-value on the F-statistic and we want this p-value to be as small as possible. This p-value is really small and this means that our regression is a good one. And notice also how there is a whole bunch of statistical information about each of the regression variables. We have the estimate, that's the coefficient, and the standard error, the t value, which is just the estimate divided by the standard error, and then the p-value. Also, notice that the estimates and the standard errors and the T-stats match exactly what we have calculated for ourselves in Excel. R makes it really easy for us to explore our data, so let's do exactly that. Let's see what happens if we regress Exxon on the

NASDAQ index, rather than on the S&P. Remember that the NASDAQ is a tech heavy index. It's dominated by stocks such as Google, Facebook, Amazon, and so on. And we can see that this regression model is not quite as good. It's R-square is only 0. 15. This is to be expected. The S&P 500 is likely to be a far better predictor of changes in Exxon than the NASDAQ. Notice also that this has a much lower F-statistic than the corresponding S&P regression. But even so, the regression coefficient of the NASDAQ is statistically significant. So let's see what happens if we include both the NASDAQ and the S&P 500 into our multiple regression. And we do so using lm with the appropriate regression specification. Here all that we need to do is to add a term for the NASDAQ on the right-hand side of the regression equation. This gives us a multiple regression with a much higher R-square. The multiple R-square here is 0. 4165. Notice that the adjusted R-square is also significantly higher than it was previously. But now here is something worth paying attention to. Check out the coefficients on the NASDAQ and the S&P 500. The NASDAQ coefficient is -1 and the S&P 500 coefficient is +1. 7. This would seem to suggest that changes in Exxon's stock are negatively correlated to changes in the NASDAQ and that does not intuitively make sense. After all, the NASDAQ and the S&P are both similar to each other, they are both US equity stock indices. Why would one of them have a large negative coefficient and the other have a large positive coefficient? Well this is a sign of a problem. The problem is multicollinearity. The whole issue here is that the NASDAQ and the S&P returns are very highly correlated with each other, so in a sense, they contain the same underlying information. And this is a tell-tale sign that our regression is not a good one, even though the R-square and the F-statistic and all of the individual T-statistics looked healthy, we have a multicollinearity problem. This can be confirmed by calculating the correlation of returns in the NASDAQ and in the S&P 500. It turns out that this correlation is over 95%, it's +0. 95. These are almost perfectly linearly correlated. This satisfies the definition of multicollinearity. We have a problem if we try to include both the NASDAQ and the S&P 500 in the same model because now both of these contain the same underlying information. This is not a reliable regression model and let's not use it. So let's go back to our data and use our data series which we intuitively think will make sense in this context and that's the price of oil. Once again, we are going to use the ETF USO as a proxy for the price of oil, and if we regress, Exxon on returns in the S&P 500 and in USO, we now get a lower R-square of 0. 328, but now both of our parameter coefficients make sense. The coefficient on the S&P 500 returns is 0. 488 and notice the 3 asterisks telling us that this is highly statistically significant. The coefficient of the USO returns is only 0. 1 and notice also that the P-value is 0. 01. R marks it with a single asterisks telling us that it statistically significant, but not very significant. So overall, this is a good regression model, it's a solid regression model with a decent R-square and also a decent adjusted R-square. It does not suffer from the problem of multicollinearity. So even though this R-square or

adjusted R-square is only 0. 32, we still prefer this model to the other which made use of both the S&P 500 and the NASDAQ, even though that model had an adjusted R-square of almost 0. 45 because that model also suffered from multicollinearity. As evidenced by the large negative and positive coefficients onto closely related variables returns on the S&P and returns on the NASDAQ.

Categorical Variables Without an Intercept in R

Let's also really quickly take a look at how we can carry out categorical regression in R. R makes it really simple because it takes care of the creation of the dummy variables. All that we need to do is keep in mind a little bit of the theory behind such regressions. Recall that there are two possible ways of setting up a regression with dummy variables. The first way is to include k-1 dummy variables and an intercept. That's one way of setting up this regression problem. If there are two groups, we would like one dummy variable and we will also retain an intercept, A1, in the regression line. That is one way of setting up such a regression. There is another way which is to have no intercept and have two dummy variables in there instead. These two ways are mathematically equivalent as you can tell from eyeballing the regression line formula. I personally prefer the method with two dummy variables instead of one just because I find it intuitively easier to interpret those coefficients A1 and A2. However, you should keep in mind that this also has some drawbacks. For instance, as we shall see in Python, the value of R-square at such a regression returns might be a little different. That's because the formula for the R-square is clearly defined only for regressions where the intercept is present. But again, mathematically, this equation is perfectly fine. For males, the dummy variable D1 will be 1 and D2 will be 0. And so, the regression line for males works out to exactly what we want. Similarly for females, the two dummy variables will have the values 0 and 1, and once again the regression line works out just fine. So given data with k groups, we can either set up k-1 dummy variables and an intercept or k dummy variables with no intercept. And in this example, I'm going to go over the second approach. Let's pick up from where we left off on Google data. Remember that we had carried out the regression of Google returns or NASDAQ returns. We have obtained an R-square of 0. 386. Let's now go ahead and see if we can test for seasonality in Google's performance relative to the NASDAQ. We are going to include the month of the year as the X variable. Now because the month of the year is categorical, it takes a value of 1 through 12. We are going to need to set up dummy variables. We could choose to set up 11 dummy variables along with an intercept or we could choose to set up 12 dummy variables and no intercept. In this case, it is literally 12 of 1 and a dozen of the other. In this case, I'm going to go with the approach of no intercept and 12 dummy

variables, so let's plow on. R makes it really easy to set up the dummy variables. We simple include a column in our data frame for the month. Notice here how we use the format function which takes in a column consisting of dates, as well as a format string, and returns a corresponding column of only the months from those dates. Let's eyeball the data. Let's type out the name of our data frame goog just to make sure that all of the months have been set up correctly and indeed they have. For instance, if the date is in December 2016, the value of the corresponding month variable is 12. For November, it's 11 and so on. And the beauty of categorical variables in R is that R's lm function is smart enough to interpret categorical data and automatically create all of the dummy variables that are required. All that you need to do is specify which of the two forms of regression you would like to go with. Here, I am instructing R to perform a regression without an intercept and I do that by including the +0 in the functional specification of the regression model. That is enough for R to conclude that it needs to set up 12 dummy variables given that there are 12 categories in the data. And this is what makes R so much easier for regression than Excel. I recall that when we had counted on this regression in Excel, we had needed to set up the 12 dummy variables along with a clever Excel formula. In any case, the regression runs through and we can see that it gives us exactly the same results as linest in Excel. Not only do we have the exact same coefficient value standard errors in t values, we also have the same R-square of 0. 49 and the same F-statistic as well. Notice that R's lm function gives us or reports the coefficients in a very convenient order, and so we can easily tell that the two statistically significant regression parameters here are those for the NASDAQ returns, that's the first one, and the second is for the month of October, that's month 10. And exactly we had discussed while carrying out the same regression in Excel, this is partially attributable to some specific months in our sample, for example, October 2008 when Google stock heavily outperformed the rest of the market. And the value of the R-square here exactly agrees with the value that we got in Excel. We are going to go ahead and repeat the same regression in Python and we shall see that we get a very slightly different value of R-square. There is a reason for that. It turns out that the R-square formula is only guaranteed to be positive and to be meaningful for regressions which include an intercept. When an intercept is not included, it is possible in some situations for the R-square to become negative. Excel and R are both fine with that. They don't change the formula of R-square because their logic is that formula is very well-known and well-excepted. Python, on the other hand, are just the value of R-square so that when the regression does not include a coefficient, it is less likely to become negative and that will account for the slight discrepancy in R-square between Excel and R, on the one hand, and Python on the other. That gets us to the end of our exploration of R as a tool for implementing regression and you should consider using R for regression no matter what your use case. In the module that's coming

up next, we will go ahead and repeat the same analysis, this time using Python. There are good reasons for also seriously considering Python for regression-style techniques. One of these is that it is a mainstream programming language which can be used in production environments. In addition, Python also helps integrate regression with a whole host of other machine-learning techniques and algorithms. We will get to that in just a little bit, but before that, let's summarize all that we learned in this module. We implemented multiple regression in R. R made it really easy for us to do so and it also made it really easy for us to interpret the results of multiple regression. And once more, we were easily able to extend our analysis to categorical variables. Regression in R is beautiful, it's simple, it's just really easy to use.

Implementing Multiple Regression Models in Python

Multiple Regression in Two Different Ways

Hello, and welcome to this module on implementing multiple regression models in Python. This is the last module of a long course. In this module, we will talk about implementing multiple regression in Python and interpreting its results. And we will also see how we can use a library known as statsmodels to build categorical analysis-based models in Python. So let's plunge right into a demo picking up right from where we left off. We had adopted one approach to regression in our previous take on regression in Python. There we had used Pandas for its data frames. We had used NumPy for its support for arrays. Scikit for the actual regression model. And then Matplotlib for the plotting bits. We want to continue with this approach, but then we will also demonstrate the use of another Python library called statsmodels for the regression. But first, let's wrap up multiple regression using scikit. Remember that Python's approach to regression, at least in the scikit version, is to treat it as a machine learning problem. A corpus is passed into a regression algorithm, that regression algorithm will find the values of the parameters, A and B. This is virtually unchanged in multiple regression as well, except that the parameters B can now be thought of as a vector rather than a single skill or value. For scikit to do its thing, it relies on the data being passed in a specific representation. The X variables are all passed in as an array of arrays with the inner array consisting of a single value. The Y values can be passed in however

we'd like. And this approach can be beautifully extended to multiple regression as well. All that we need to do is to extend each element of the X array such that there are multiple X coordinates in each inner element. The Y values are completely unchanged and the rest of the regression functions the same as before. And I'm going to breeze through the first part of this regression setup because it's identical to the version we discussed at length in simple regression. We set up a bunch of variables containing the file names and we also set up a function called readFile which reads in the contents of those file names, converts the prices into returns, and gives them back to us in a nicely setup data frame. With that out of the way, we can actually now set up the regression. So we have moved onto the scikit parts. Here we are going to regress the returns of Exxon on the NASDAQ and on the ETF USO. As usual, when we convert from prices to returns, we are going to effectively lose one data point from our data, so let's also go ahead and exclude that data. This is to make sure that we don't get any problems when we go ahead with our regression later on. The one bit worth commenting on here is the fact that the oil ETF has its data starting three months after the NASDAQ data, and so we've got to make sure that we handle this missing data appropriately. The way we do this is to include only those values of NASDAQ, which also have a corresponding value for the oil returns. So we restrict the NASDAQ array to the length of the oil ETF array and we create a combined X variable. This is the combined matrix that you see on screen now. Notice how we make use of the np. vstack method. As its name would suggest, if you pass in two NumPy arrays into np. vstack, it stacks them vertically. Here, we actually would like them to be stacked horizontally rather than vertically, so we simply calculate the transpose using the. T operation. And as you can see now, we have our X data nicely set up in exactly the format that we'd like. We have an array of arrays. Each inner element consists of two values corresponding to the two X variables in our regression. The Y data can be used exactly as-is from the data frame. This we obtain by reading in data from the Exxon mobile data file. We restrict it to only those values for which we have values of both of our regressors and then we instantiate the linear regression object and call the fit method on it. This is it. The regression has been performed and we can calculate the residuals and the R-square and so on. For instance, the R-square of this is 0. 23, which exactly agrees with the value that we got in Excel and in R. Now this approach, the scikit approach to linear regression treats it mostly as a machine learning problem rather than a statistical problem. This makes it hard for us to get detailed diagnostics of the linear model that we have fitted. This is something that we need to fix, particularly with multiple regression, where as we've seen the regression statistics are very important to analyze. And we can do this easily making use of a different approach, a different Python library called statsmodels. So let's start by importing statsmodels. This is going to give us a way to check out the results of our regression and as much detail as in R and it's very, very simple indeed to use.

One important little detail to keep in mind, by default, the statsmodel linear regression does not include a constant, so we've got to explicitly add a constant into our data. We can do this by using the statsmodel. api. add_constant method. We pass in our pre-existing X data and what we get back is a new version of the X data with one value tacked on. This is the value one for every element of our nested array. And that really is all there is to it. We simply fit the regression model using the. ols method where we specify the Y and the X values. We get back a model object, we invoke the fit method on the this model object, and then we go ahead and print out the results of that fit and a wealth of regression information appears before us. This is just as detailed and as comprehensive, in fact, more detailed and more comprehensive than the information that we get in R. This is a familiar result because we have seen it both with linest in Excel and with lm in R. The R-square of this regression is 0. 229. The adjusted R-square is 0. 217. We have a constant, as well as two explanatory variables. The coefficients of those X variables exactly agree with our previous versions in Excel and R. There are also standard errors, t values, p-values, F-statistics, and even more. Notice, for instance, that the skewness and the kurtosis of the residuals are also reported. There are a bunch of other tests, Durbin-Watson and Jarque-Bera. These are as for more advanced statistical properties which are beyond the scope of this course, but you get the idea. Python also has support for rich regression information and this is in the stacks model, I believe, which is really easy to use.

Categorical Variables

Let's round off this module with a look at categorical variables and how we can use them with regression in Python. As we shall see, this is extremely easy to do in Python; however, we should be careful while interpreting the results of a regression which does not have an intercept because the R-square reported by stacks model will be ever so slightly different from that reported in Excel or in R. Let's carry out the same categorical regression that we did in Excel and in R. We are going to regress Google's returns against the NASDAQ returns, but with additional variables for any affect related to the month of the year. So here, we need to create a categorical variable which has the month of the year of a particular data point. This is easy enough for us to do because we have a column in our pandas data frame containing the date stored as a string. All we need to do is extract the appropriate characters from this date string and convert it into the month. So let's do exactly that and add that as a column back into our pandas data frame. We add here a month column to the data frame that includes the month, which has been extracted from the date column. That's character's five through seven of the date column. They've been converted into an integer. And so, for a data point in December 2016, the month column will be 12,

for November it will 11, and so on and so forth. Now this column contains categorical data because the data here takes a finite set of values 1 through 12. We need to convert this into a set of dummy variables. In R, this was taken care of for us by lm. We did not explicitly need to clear dummy variables, but in Python, we do. However, this is simple enough for us to do. We simply make use of the method, sm. categorical. We specify the data, which in this case is the column from our months column, but we do need to reshape it first. And notice that we also tell statsmodel to drop that categorical variable using the drop equal to true parameter there. Otherwise, the original variable would have been included in our dummy variable data. And that gives us a perfect set of dummy variables. There are 12 such in every row and we have 1 such row for every point in the original data. This is perfect for us to use in our regression. We still have a little bit of data manipulation to do because the exact form of the different X and Y variables needs to be lined up perfectly. Take, for instance, the other X variable that is returns from the NASDAQ, this is still in the form of a pandas data frame, and more importantly, this still has one empty element at the end, which is nan. If we check out the shape of this NumPy array, we see that it is a data frame, that's why the second coordinate here is blank. If we also invoke the shape method on our dummy matrix, we will find that it has a different shape, so our first order of business is to line up the NASDAQ data with the dummy variables. This we do using the reshape method and we can confirm that this does indeed give us the same shape or a similar shape to our dummy data. Now we still have one missing value for the very first data point. This has to do with the conversion from prices to returns. In order to ensure that this does not get in the way of our regression, let's eliminate this data point, so we will exclude it in Python. A negative index counts from the end of an array, and so effectively, we are skipping the last element in our array. And then stack all of them nicely using the np. hstack method. Specify two parameters, the first is our dummy matrix, the second is our NASDAQ data reshaped to be in the right format. That forms the X data for use in our regression. It's always a lot easier to set up the Y data. That is simply Googles returns reshaped appropriately and with the last row eliminated as before. And with that, we are good to go. We have both our X data and our Y data set up correctly. In this instance, we do not want to add a constant to our X data. Remember that we are carrying out the categorical regression. We have 12 groups in our data and we have chosen to go with 12 dummy variables and no intercept. So we just go ahead and invoke the correct methods from statsmodel. And the results up here before us, these almost exactly match the results of the same regression performed in Excel and in R. Notice the values of the coefficients, the standard errors, and the t-statistics are all exactly the same. For instance, the NASDAQ data is the most statistically significant with a t-stat of 8. 927. The dummy variable for October is next with a t-stat of 3. 649. This is exactly the same as we saw in R, but notice that the R-square is just a little bit different and the reason for this has to do

with the absence of an intercept in our regression. As we've already discussed, the R-square is only well-defined and guaranteed to be positive if the regression includes an intercept. When the intercept is not included, the R-square formula is not clearly defined. Excel and R both take one approach in such situations. Python takes a slightly different approach and that is why there is a very slight difference in the values of the R-square across these technologies, but we know that we've carried out this regression correctly because all of the intercept coefficients and other statistics perfectly match. We have successfully carried out regression using categorical variables in Python. That brings us to the end of this module and indeed to the end of this course. So let's really quickly go back and discuss the few points, which I hope you'll remember even if you forget everything else. The most common use case of regression is for connecting the dots. Simple regression involves dots in two dimensions, multiple regression, intervals, dots, and multiple dimensions. Regression is ideally suited to two common applications. The first of these is explaining variance. The second of these is making predictions from data which we already have. Regression is a great tool because it's powerful, versatile, and deep. We also implemented regression in Excel, R, and in Python. You should seriously consider using R for regression whatever your use case, but Excel and Python have their important benefits as well. There is a lot that you can do with your newly acquired knowledge of regression. You can use it to make your analysis in corporate presentations much more polished, you can go deep into research topics, or you can apply new algorithms in areas such as machine learning. So that's where we can go from here. Let's summarize all that we've just covered. In this module, we focused on implementing multiple regression in Python and we did so using are old approach, making use of scikit and also a new approach making use of statsmodel. We saw how to interpret the wealth of regression information that statsmodel provides us. And we also carried out multiple regression including categorical variables. We learned how we've got to exercise care while interpreting the R-square of a regression which does not have an intercept.

Course author

Vitthal Srinivasan

Vitthal has spent a lot of his life studying - he holds Masters Degrees in Math and Electrical Engineering from Stanford, an MBA from INSEAD, and a Bachelors Degree in Computer Engineering from...

Course info

Level	Beginner
Rating	★★★★★ (50)
My rating	★★★★★
Duration	4h 12m
Released	10 Feb 2017

Share course