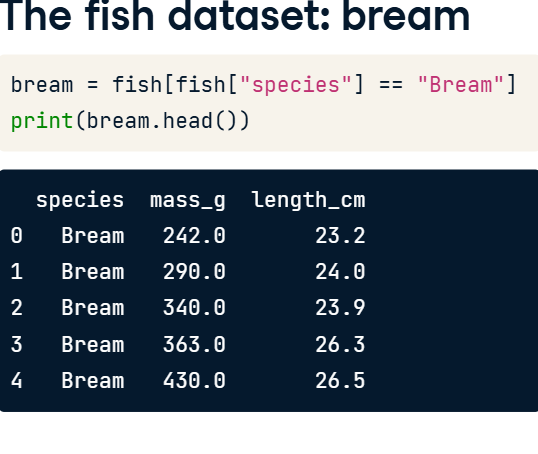
**Making predictions**

The big benefit of running models rather than simply calculating descriptive statistics is that models let you make predictions.**The fish dataset: bream**

Here's the fish dataset again. This time, we'll look only at the bream data. There's a new explanatory variable too: the length of each fish, which we'll use to predict the mass of the fish.

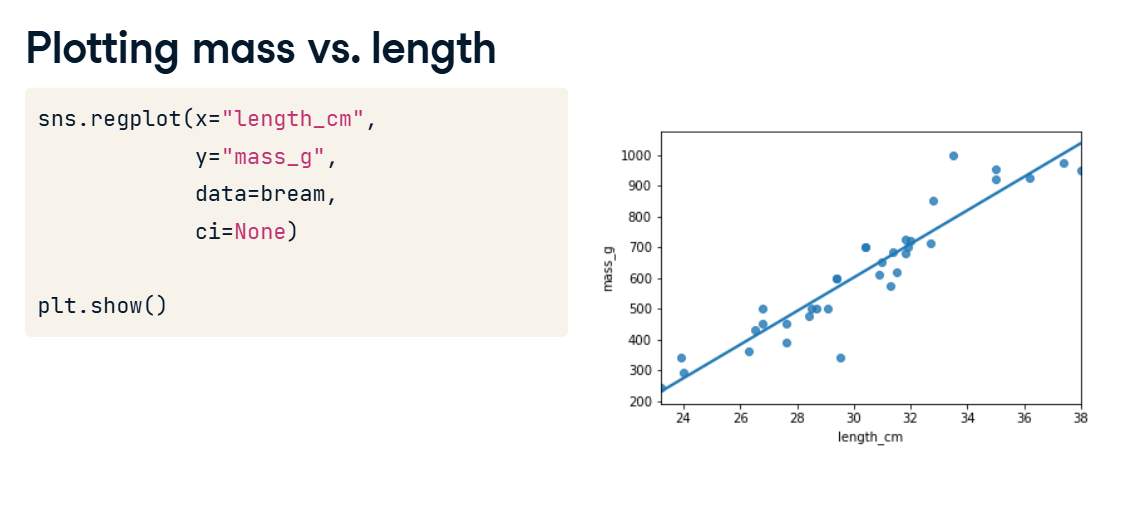


**3. Plotting mass vs. length**

Here's a scatter plot of mass versus length for the bream data, with a linear trend line.

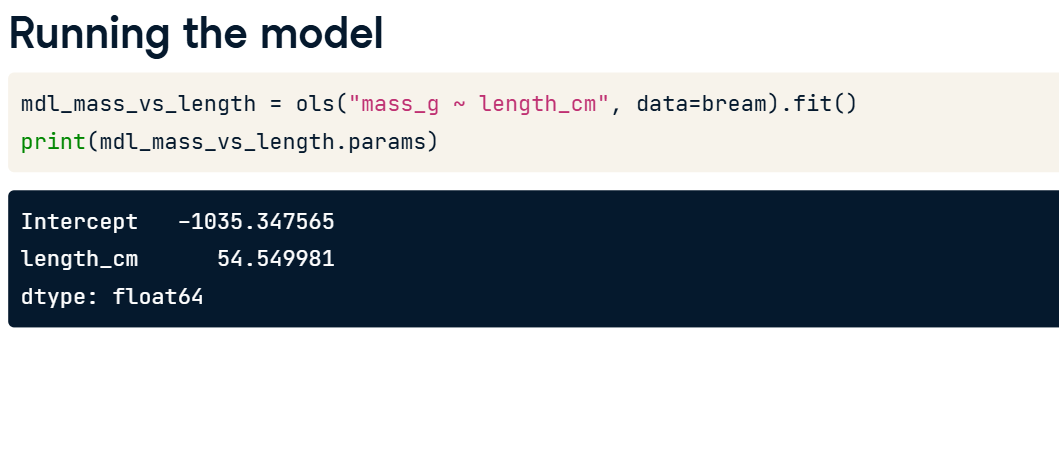
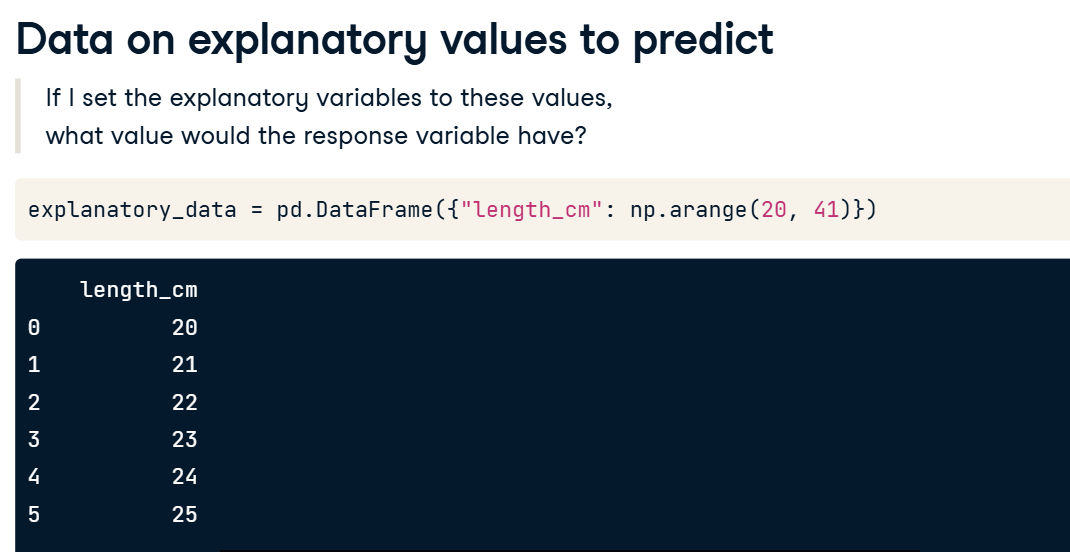
**Running the model**

Before we can make predictions, we need a fitted model. As before, we call ols with a formula and the dataset, after which we add dot fit. The response, mass in grams, goes on the left-hand side of the formula, and the explanatory variable, length in centimeters, goes on the right. We need to assign the result to a variable to reuse later on. To view the coefficients of the model, we use the params attribute in a print call.



**5. Data on explanatory values to predict**

The principle behind predicting is to ask questions of the form "if I set the explanatory variables to these values, what value would the response variable have?". That means that the next step is to choose some values for the explanatory variables. To create new explanatory data, we need to store our explanatory variables of choice in a pandas DataFrame. You can use a dictionary to specify the columns. For this model, the only explanatory variable is the length of the fish. You can specify an interval of values using the np dot arange function, taking the start and end of the interval as arguments. Notice that the end of the interval does not include this value. Here, I specified a range of twenty to forty centimeters.



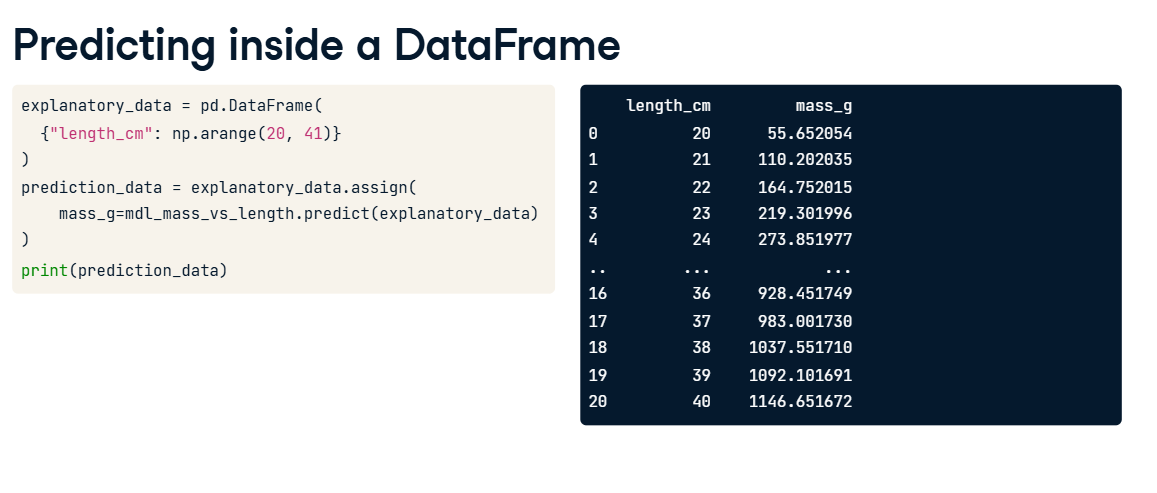
**6. Call predict()**

The next step is to call predict on the model, passing the DataFrame of explanatory variables as the argument. The predict function returns a Series of predictions, one for each row of the explanatory data.



**7. Predicting inside a DataFrame**

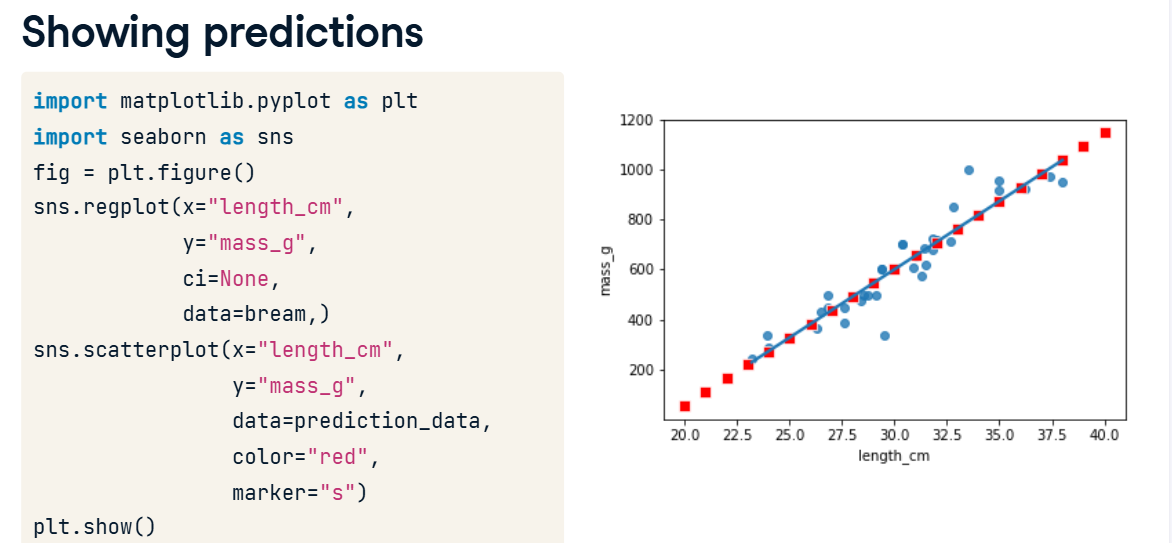
Having a single column of predictions isn't that helpful to work with. It's easier to work with if the predictions are in a DataFrame alongside the explanatory variables. To do this, you can use the pandas assign method. It returns a new object with all original columns in addition to new ones. You start with the existing column, explanatory\_data. Then, you use dot assign to add a new column, named after the response variable, mass\_g. You calculate it with the same predict code from the previous slide. The resulting DataFrame contains both the explanatory variable and the predicted response. Now we can answer questions like "how heavy would we expect a bream with length twenty three centimeters to be?", even though the original dataset didn't include a bream of that exact length. Looking at the prediction data, you can see that the predicted mass is two hundred and nineteen grams.



**8. Showing predictions**

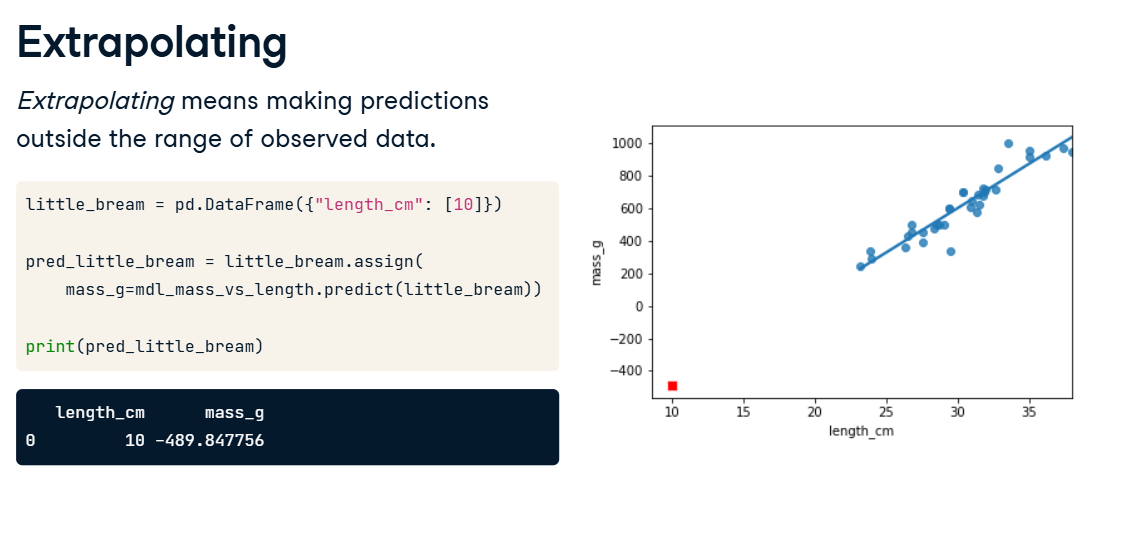
03:09 - 03:39

Let's include the predictions we just made on the scatter plot. To plot multiple layers, we set a matplotlib figure object called fig before calling regplot and scatterplot. As a result, the plt dot show call will then plot both graphs on the same figure. I've marked the prediction points in red squares to distinguish them from the actual data points. Notice that the predictions lie exactly on the trend line.



**9. Extrapolating**

All the fish were between twenty three and thirty eight centimeters, but the linear model allows us to make predictions outside that range. This is called extrapolating. Let's see what prediction we get for a ten centimeter bream. To achieve this, you first create a DataFrame with a single observation of 10 cm. You then predict the corresponding mass as before. Wow. The predicted mass is almost minus five hundred grams! This is obviously not physically possible, so the model performs poorly here. Extrapolation is sometimes appropriate, but can lead to misleading or ridiculous results. You need to understand the context of your data in order to determine whether it is sensible to extrapolate.

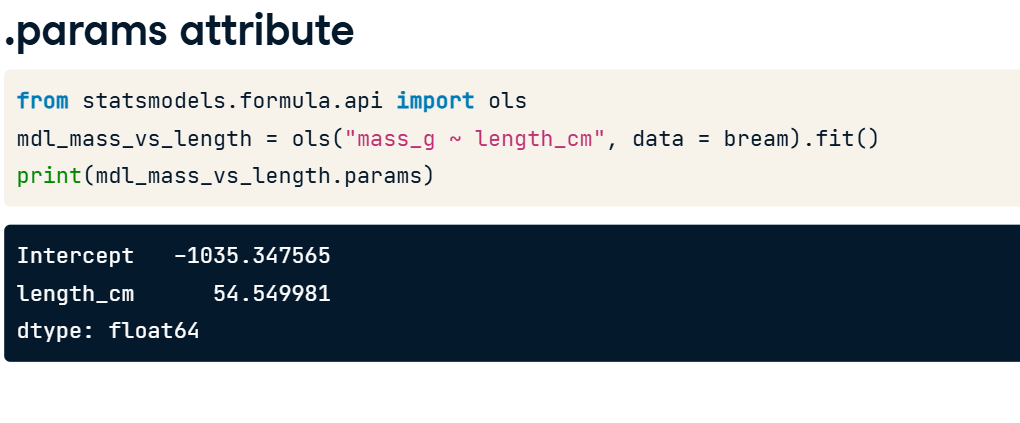


**Working with model objects**

The model objects created by ols contain a lot of information. In this video, you'll see how to extract it.

**2. .params attribute**

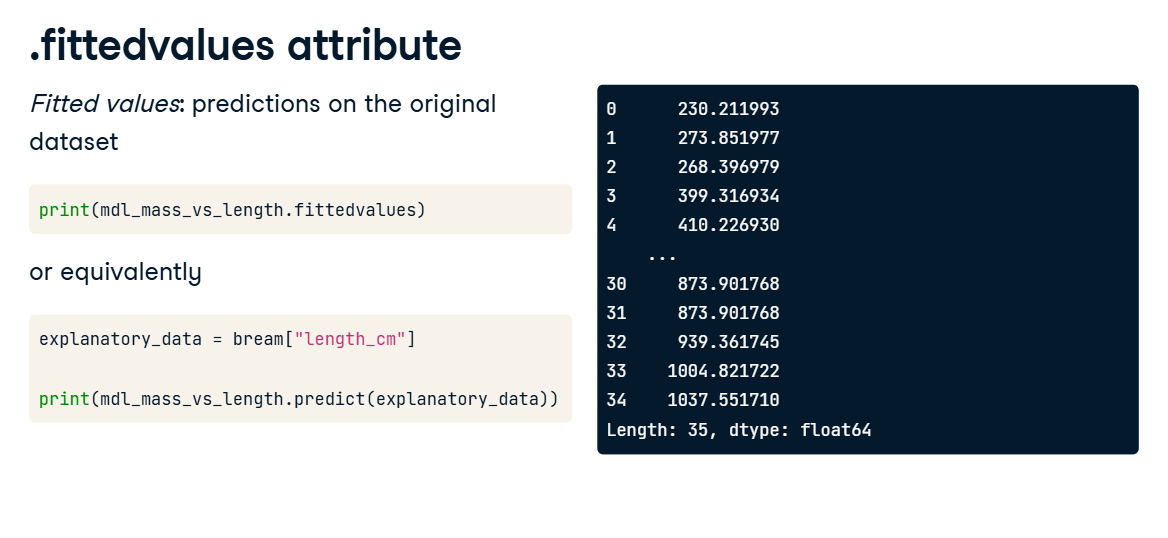
You already learned how to extract the coefficients or parameters from your fitted model. You add the dot params attribute, which will return a pandas Series including your intercept and slope.



**3. .fittedvalues attribute**

00:21 - 00:48

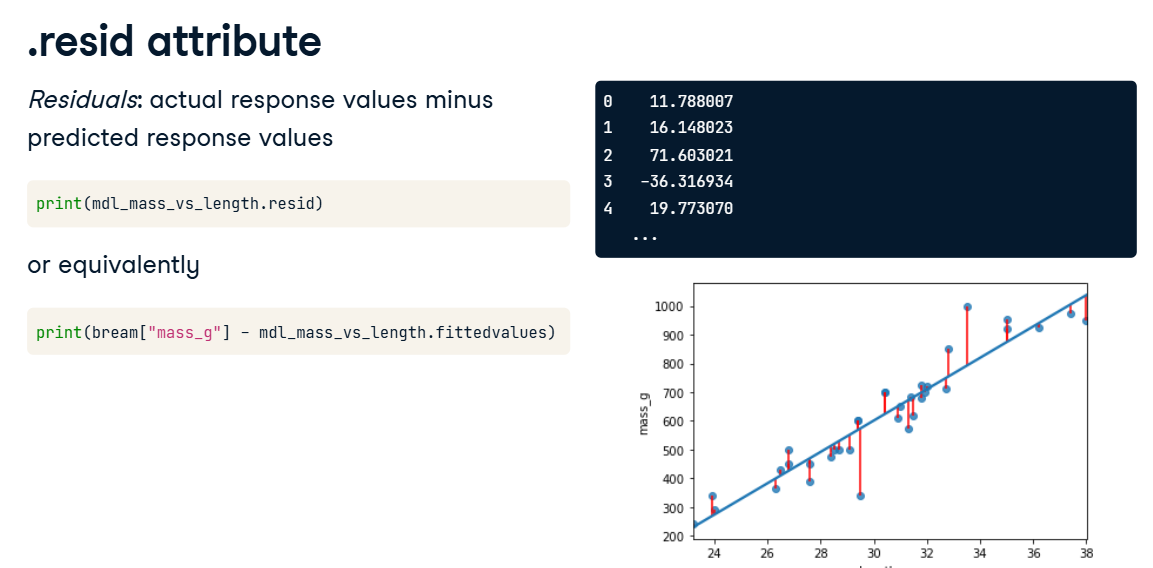
"Fitted values" is jargon for predictions on the original dataset used to create the model. Access them with the fittedvalues attribute. The result is a pandas Series of length thirty five, which is the number of rows in the bream dataset. The fittedvalues attribute is essentially a shortcut for taking the explanatory variable columns from the dataset, then feeding them to the predict function.



**4. .resid attribute**

00:48 - 01:28

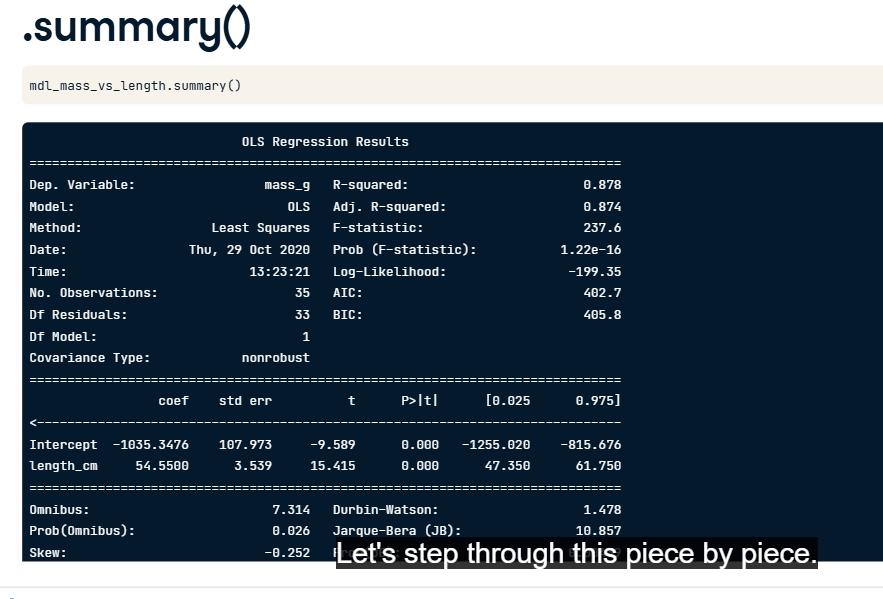
"Residuals" are a measure of inaccuracy in the model fit, and are accessed with the resid attribute. Like fitted values, there is one residual for each row of the dataset. Each residual is the actual response value minus the predicted response value. In this case, the residuals are the masses of breams, minus the fitted values. I illustrated the residuals as red lines on the regression plot. Each vertical line represents a single residual. You'll see more on how to use the fitted values and residuals to assess the quality of your model in Chapter 3.



**5. .summary()**

01:28 - 01:37

The summary method shows a more extended printout of the details of the model. Let's step through this piece by piece.



**6. .summary() part 1**

01:37 - 01:50

First, you see the dependent variable(s) that were used in the model, in addition to the type of regression. You also see some metrics on the performance of the model. These will be discussed in the next chapter.

**7. summary() part 2**

In the second part of the summary, you see details of the coefficients. The numbers in the first column are the ones contained in the params attribute. The numbers in the fourth column are the p-values, which refer to statistical significance. You can learn about them in DataCamp's courses on inference. The last part of the summary are diagnostic statistics that are outside the scope of this course.

**Regression to the mean**

Let's take a short break from thinking about regression modeling, to a related concept called "regression to the mean". Regression to the mean is a property of the data, not a type of model, but linear regression can be used to quantify its effect.

**The concept**

You already saw that each response value in your dataset is equal to the sum of a fitted value, that is, the prediction by the model, and a residual, which is how much the model missed by. Loosely speaking, these two values are the parts of the response that you've explained why it has that value, and the parts you couldn't explain with your model. There are two possibilities for why you have a residual. Firstly, it could just be because your model isn't great. Particularly in the case of simple linear regression where you only have one explanatory variable, there is often room for improvement. However, it usually isn't possible or desirable to have a perfect model because the world contains a lot of randomness, and your model shouldn't capture that. In particular, extreme responses are often due to randomness or luck. That means that extremes don't persist over time, because eventually the luck runs out. This is the concept of regression to the mean. Eventually, extreme cases will look more like average cases.

**Pearson's father son dataset**

Here's a classic dataset on the heights of fathers and their sons, collected by Karl Pearson, the statistician who the Pearson correlation coefficient is named after. The dataset consists of over a thousand pairs of heights, and was collected as part of a nineteenth century scientific work on biological inheritance. It lets us answer the question, "do tall fathers have tall sons?", and "do short fathers have short sons?".

1. 1 Adapted from https://www.rdocumentation.org/packages/UsingR/topics/father.son

**Scatter plot**

Here's a scatter plot of the sons' heights versus the fathers' heights. I've added a line where the son's and father's heights are equal, using plt dot axline. The first two arguments determine the intercept and slope, while the linewidth and color arguments help it stand out. I also used plt dot axis with the 'equal' argument so that one centimeter on the x-axis appears the same as one centimeter on the y-axis. If sons always had the same height as their fathers, all the points would lie on this green line.

**5. Adding a regression line**

02:25 - 02:55

Let's add a black linear regression line to the plot using regplot. You can see that the regression line isn't as steep as the first line. On the left of the plot, the black line is above the green line, suggesting that for very short fathers, their sons are taller than them on average. On the far right of the plot, the black line is below the green line, suggesting that for very tall fathers, their sons are shorter than them on average.

**6. Running a regression**

02:55 - 03:08

Running a model quantifies the predictions of how much taller or shorter the sons will be. Here, the sons' heights are the response variable, and the fathers' heights are the explanatory variable.

**Making predictions**

Now we can make predictions. Consider the case of a really tall father, at one hundred and ninety centimeters. At least, that was really tall in the late nineteenth century. The predicted height of the son is one hundred and eighty-three centimeters. Tall, but not quite as tall as his dad. Similarly, the prediction for a one hundred and fifty-centimeter father is one hundred and sixty-three centimeters. Short, but not quite as short as his dad. In both cases, the extreme value became less extreme in the next generation — a perfect example of regression to the mean.

**Transforming variables**

Sometimes, the relationship between the explanatory variable and the response variable may not be a straight line. To fit a linear regression model, you may need to transform the explanatory variable or the response variable, or both of them.

**2. Perch dataset**

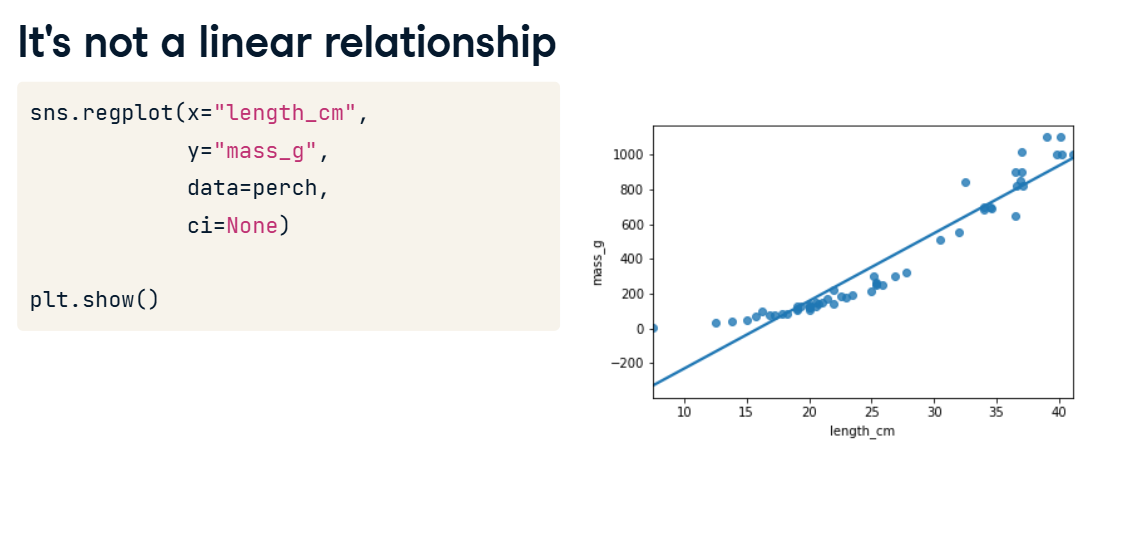
00:15 - 00:18

Consider the perch in the fish dataset.

**3. It's not a linear relationship**

00:18 - 00:25

The upward curve in the mass versus length data prevents us drawing a straight line that follows it closely.



**4. Bream vs. perch**

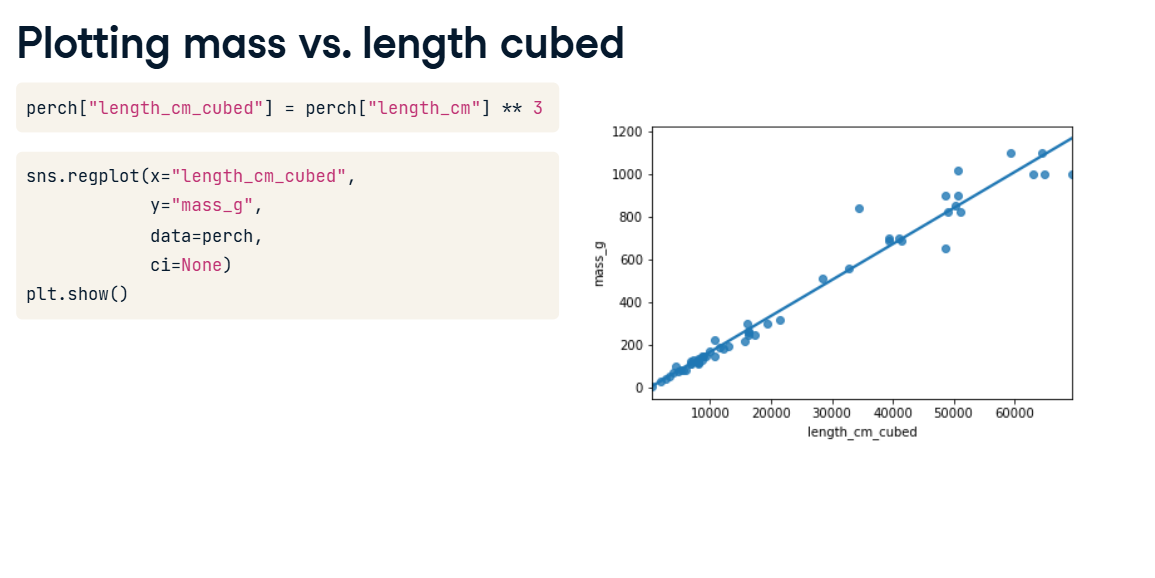
00:25 - 01:02

To understand why the bream had a strong linear relationship between mass and length, but the perch didn't, you need to understand your data. I'm not a fish expert, but looking at the picture of the bream on the left, it has a very narrow body. I guess that as bream get bigger, they mostly get longer and not wider. By contrast, the perch on the right has a round body, so I guess that as it grows, it gets fatter and taller as well as longer. Since the perches are growing in three directions at once, maybe the length cubed will give a better fit.

**5. Plotting mass vs. length cubed**

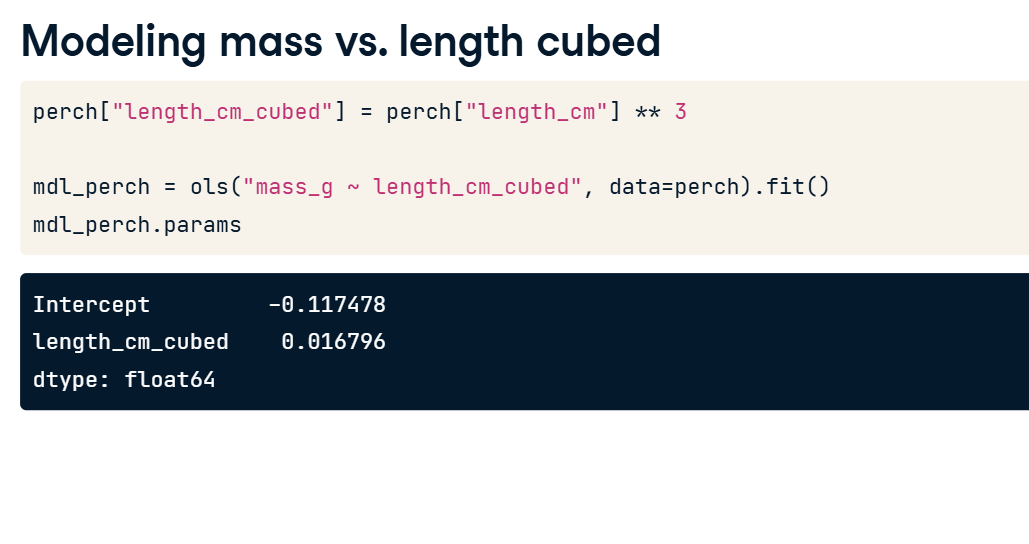
01:02 - 01:24

Here's an update to the previous plot. The only change is that the x-axis is now length to the power of three. To do this, first create an additional column where you calculate the length cubed. Then replace this newly created column in your regplot call. The data points fit the line much better now, so we're ready to run a model.



**6. Modeling mass vs. length cubed**

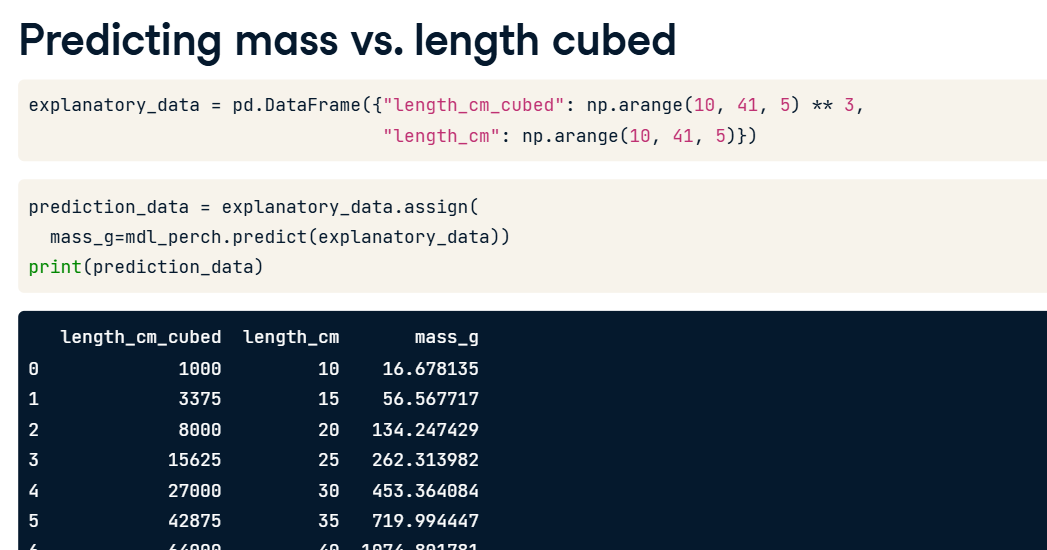
01:24 - 01:35

To model this transformation, we replace the original length variable with the cubed length variable. We then fit the model and extract its coefficients. 

**7. Predicting mass vs. length cubed**

01:35 - 01:53

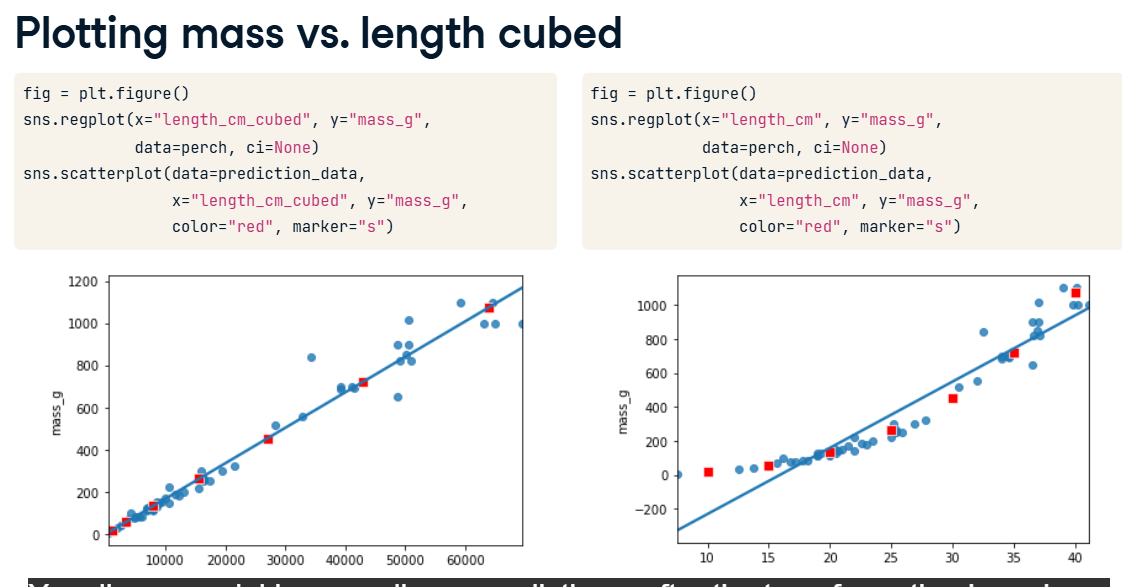
We create the explanatory DataFrame in the same way as usual. Notice that you specify the lengths cubed. We can also add the untransformed lengths column for reference. The code for adding predictions is the same assign and predict combination as you've seen before.



**8. Plotting mass vs. length cubed**

01:53 - 02:19

The predictions have been added to the plot of mass versus length cubed as red points. As you might expect, they follow the line drawn by regplot. It gets more interesting on the original x-axis. Notice how the red points curve upwards to follow the data. Your linear model has non-linear predictions, after the transformation is undone.



**9. Facebook advertising dataset**

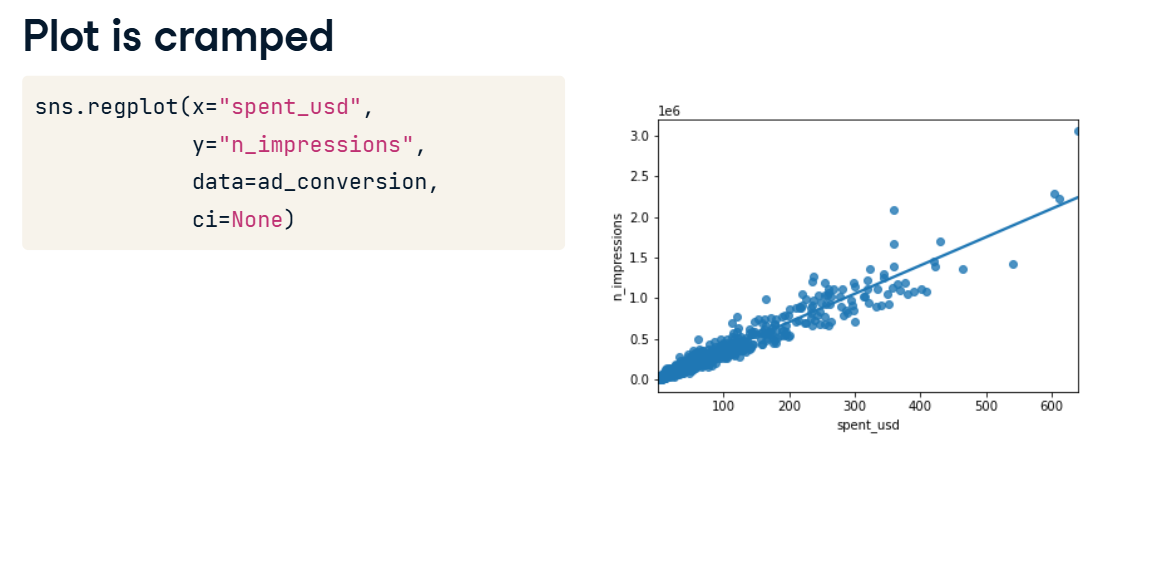
02:19 - 02:38

Let's try one more example using a Facebook advertising dataset. The flow of online advertising is that you pay money to Facebook, who show your advert to Facebook users. If a person sees the advert, it's called an impression. Then some people who see the advert will click on it.

**10. Plot is cramped**

02:38 - 02:51

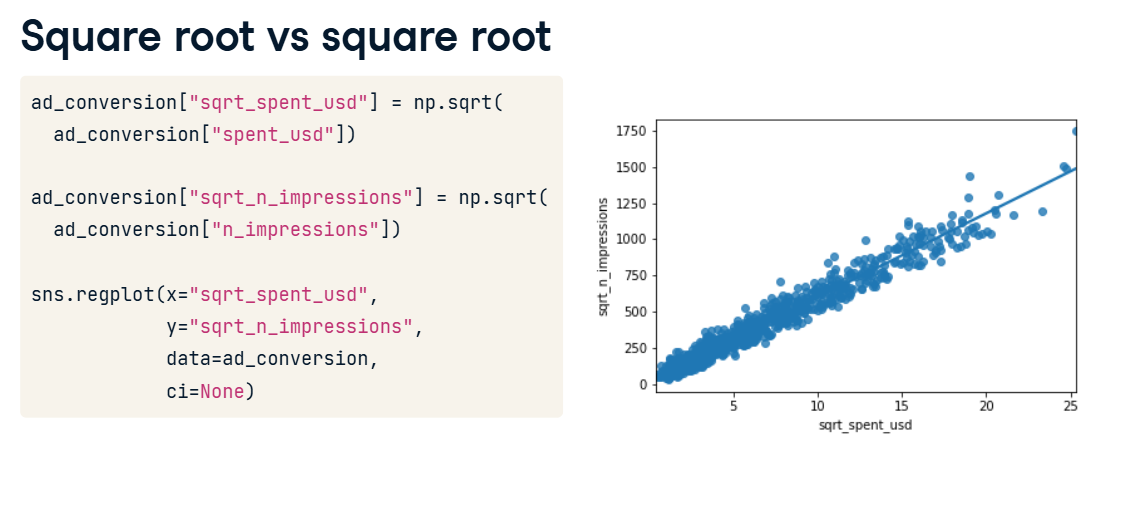
Let's look at impressions versus spend. If we draw the standard plot, the majority of the points are crammed into the bottom-left of the plot, making it difficult to assess whether there is a good fit or not.



**11. Square root vs square root**

02:51 - 03:07

By transforming both the variables with square roots, the data are more spread out throughout the plot, and the points follow the line fairly closely. Square roots are a common transformation when your data has a right-skewed distribution.



**12. Modeling and predicting**

Running the model and creating the explanatory dataset are the same as usual, but notice the use of the transformed variables in the formula and DataFrame. I also included the untransformed spent\_usd variable for reference. Prediction requires an extra step. Because we took the square root of the response variable (not just the explanatory variable), the predict function will predict the square root of the number of impressions. That means that we have to undo the square root by squaring the predicted responses. Undoing the transformation of the response is called back transformation.

