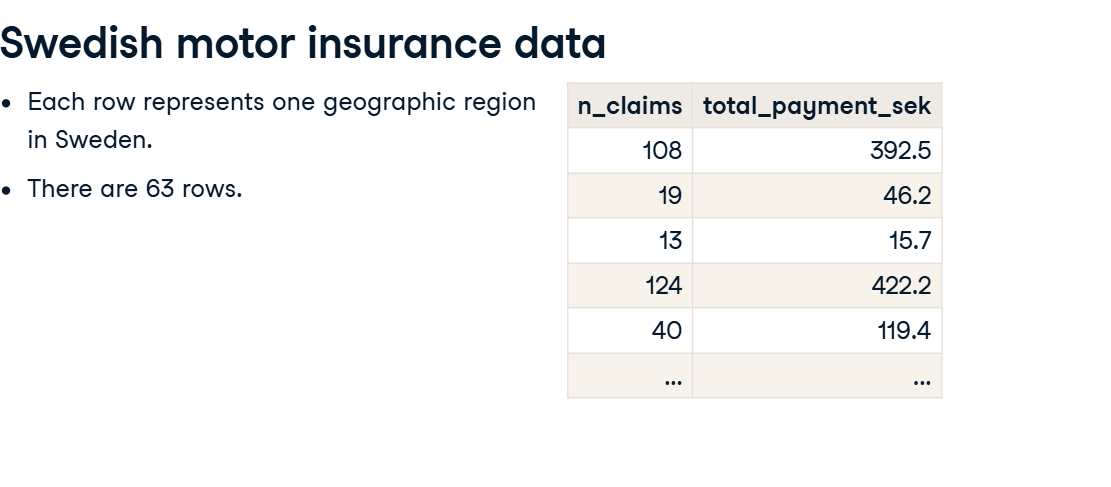
**A tale of two variables**

Hi, my name is Maarten and welcome to the course. You will be learning about regression, a statistical tool to analyze the relationships between variables. Let's start with an example.

**Swedish motor insurance data**

This dataset on Swedish motor insurance claims is as simple as it gets. Each row represents a region in Sweden, and the two variables are the number of claims made in that region, and the total payment made by the insurance company for those claims, in Swedish krona.



**Descriptive statistics**

This course assumes you have experience with calculating descriptive statistics on variables in a DataFrame. For example, calculating the mean of each variable. We can use pandas for this, as shown here. The course also assumes you understand the correlation between two variables. Here, the correlation is 0 point nine one, a strong positive correlation. That means that as the number of claims increases, the total payment typically increases as well.



**What is regression?**

Regression models are a class of statistical models that let you explore the relationship between a response variable and some explanatory variables. That is, given some explanatory variables, you can make predictions about the value of the response variable. In the insurance dataset, if you know the number of claims made in a region, you can predict the amount that the insurance company has to pay out. That lets you do thought experiments like asking how much the company would need to pay if the number of claims increased to two hundred.

**Jargon**

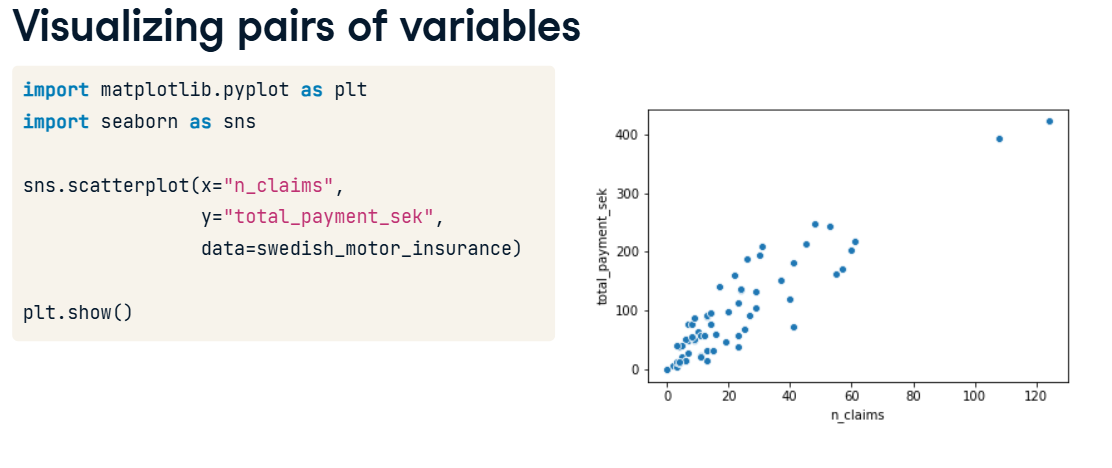
The response variable, the one you want to make predictions on, is also known as the dependent variable or the y variable. These two terms are completely interchangeable. Explanatory variables, used to explain how the predictions will change, are also known as independent variables or x variables. Again, these terms are interchangeable.

**Linear regression and logistic regression**

In this course we're going to look at two types of regression. Linear regression is used when the response variable is numeric, like in the motor insurance dataset. Logistic regression is used when the response variable is logical. That is, it takes True or False values. We'll limit the scope further to only consider simple linear regression and simple logistic regression. This means you only have a single explanatory variable.

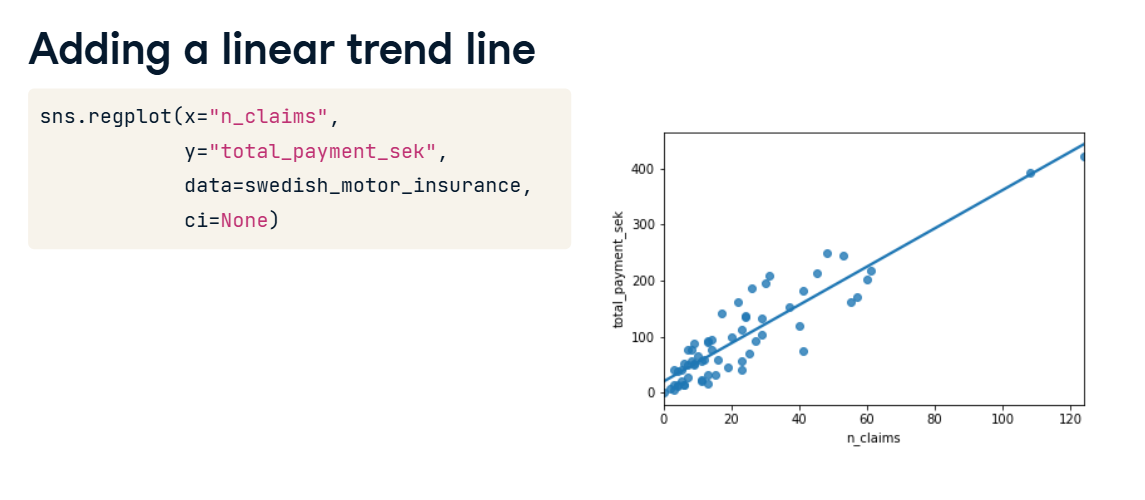
**Visualizing pairs of variables**

Before you start running regression models, it's a good idea to visualize your dataset. To visualize the relationship between two numeric variables, you can use a scatter plot. The course assumes that your data visualization skills are strong enough that you can understand the seaborn code written here. If not, try taking one of DataCamp's courses on seaborn before you begin this course. On the plot, you can see that the total payment increases as the number of claims increases. It would be nice to be able to describe this increase more precisely.



**Adding a linear trend line**

One refinement we can make is to add a trend line to the scatter plot. A trend line means fitting a line that follows the data points. In seaborn, trend lines are drawn using the regplot() function, which adds a trend line calculated using linear regression. By default, regplot() adds a confidence interval around the line, which we can remove by setting the ci argument to None. The trend line is mostly quite close to the data points, so we can say that the linear regression is a reasonable fit.



**Course flow**

Here's the plan for the course. First, we'll visualize and fit linear regressions. Then we'll make predictions with them. Thirdly, we'll look at ways of quantifying whether or not the model is a good fit. In the final chapter, we'll run through this flow again using logistic regression models.

**Python packages for regression**

Before we dive into the first exercise, a word on Python packages for regression. Both statsmodels and scikit-learn can be used. However, statsmodels is more optimized for insight, whereas scikit-learn is more optimized for prediction. Since we'll focus on insight, we'll be using statsmodels in this course.

**Fitting a linear regression**

You may have noticed that the linear regression trend lines in the scatter plots were straight lines. That's a defining feature of a linear regression.

**2. Straight lines are defined by two things**

Straight lines are completely defined by two properties. The intercept is the y value when x is zero. The slope is the steepness of the line, equal to the amount y increases if you increase x by one. The equation for a straight line is that the y value is the intercept plus the slope times the x value.

**3. Estimating the intercept**

Here's the trend line from the Swedish insurance dataset. Let's try to estimate the intercept.

To find the intercept, look at where the trend line intersects the y axis.

Its less than half way to the fifty mark, so I'd guess it's about twenty.

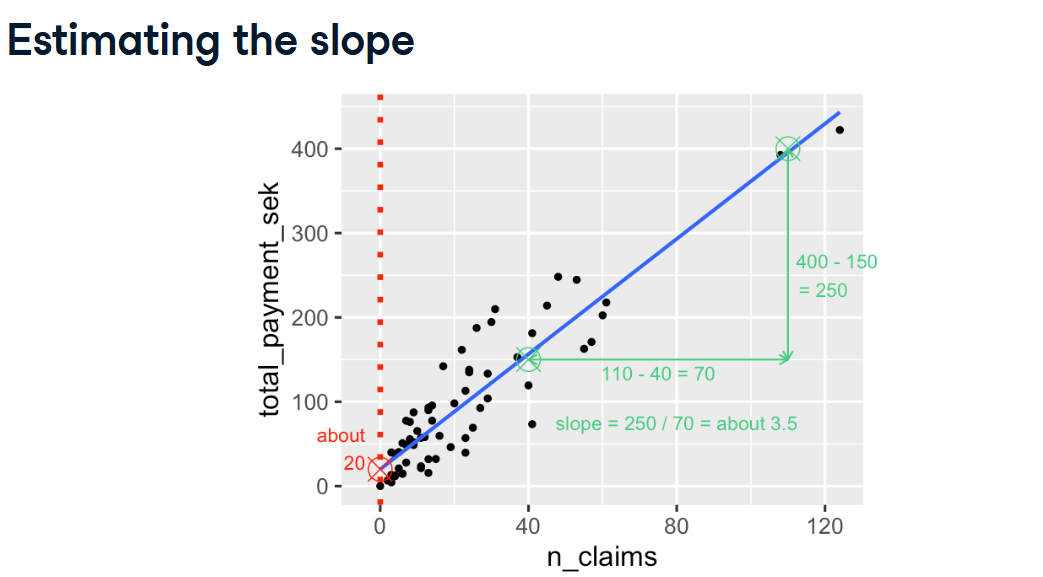
**Estimating the slope**

To estimate the slope, we need two points. To make the guessing easier, I've chosen points where the line is close to the gridlines.

First, we calculate the change in y values between the points. One y value is about four hundred and the other is about one hundred and fifty, so the difference is two hundred and fifty.

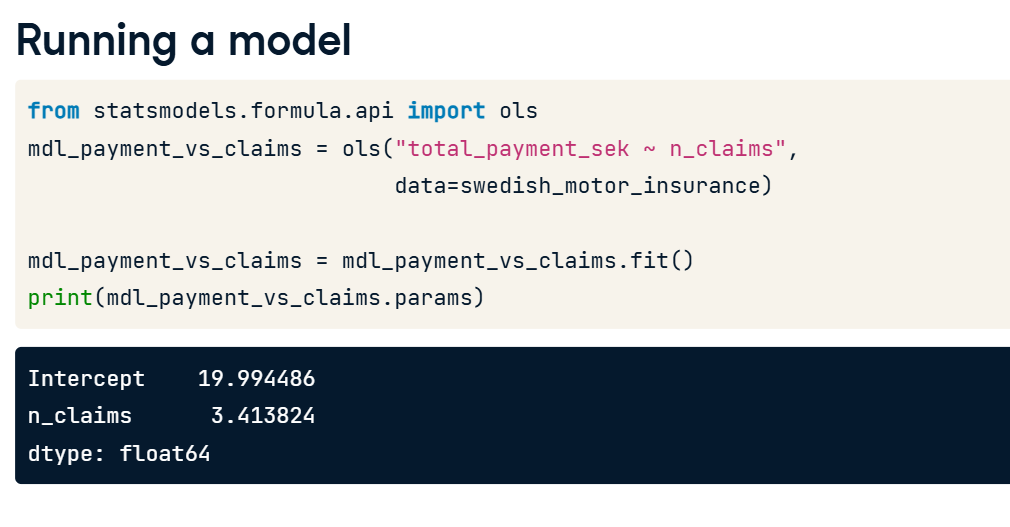
Now we do the same for the x axis. One point is at one hundred and ten, the other at forty. So the difference is seventy.

To estimate the slope we divide one number by the other. Two hundred and fifty divided by seventy is about three point five, so that is our estimate for the slope. Let's run a linear regression to check our guess.



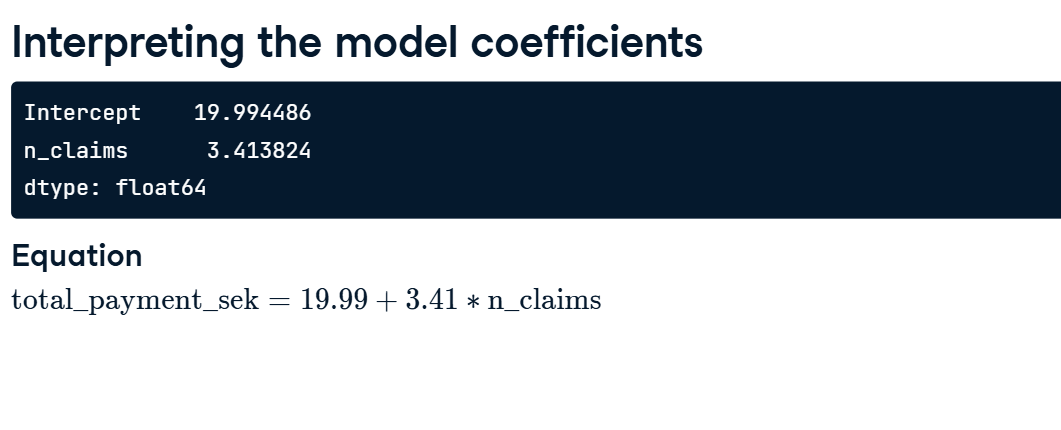
**Running a model**

To run a linear regression model, you import the ols function from statsmodels dot formula dot api. OLS stands for ordinary least squares, which is a type of regression, and is commonly used. The function ols takes two arguments. The first argument is a formula: the response variable is written to the left of the tilde, and the explanatory variable is written to the right. The data argument takes the DataFrame containing the variables. To actually fit the model, you add the dot fit() method to your freshly created model object. When you print the resulting model, it's helpful to use the params attribute, which contains the model's parameters. This will result in two coefficients. These coefficients are the intercept and slope of the straight line. It seems our guesses were pretty close. The intercept is very close to our estimate of twenty. The slope, indicated here as n\_claims, is three point four, slightly lower than what we guessed.



**Interpreting the model coefficients**

That means that we expect the total payment to be twenty plus three point four times the number of claims. So for every additional claim, we expect the total payment to increase by three point four.



**1. Categorical explanatory variables**

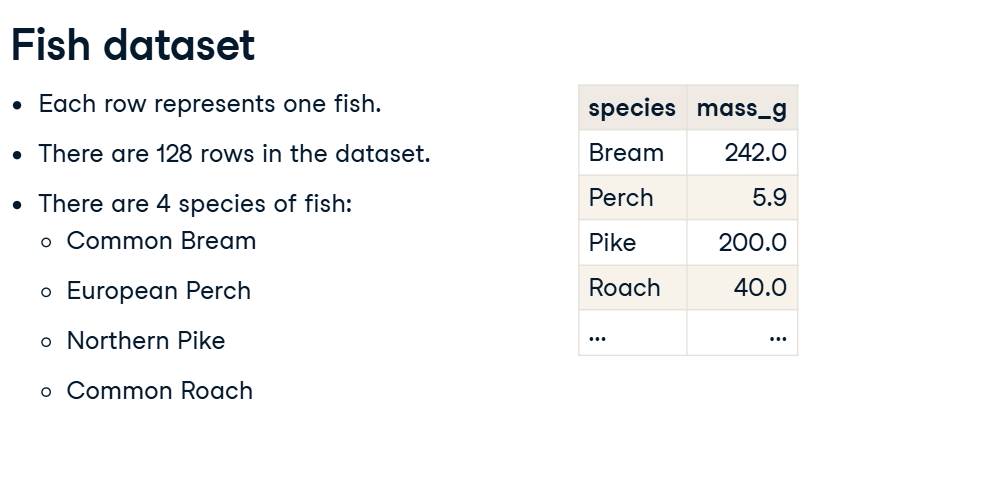
00:00 - 00:10

So far we looked at running a linear regression using a numeric explanatory variable. Now let's look at what happens with a categorical explanatory variable.

**2. Fish dataset**

00:10 - 00:23

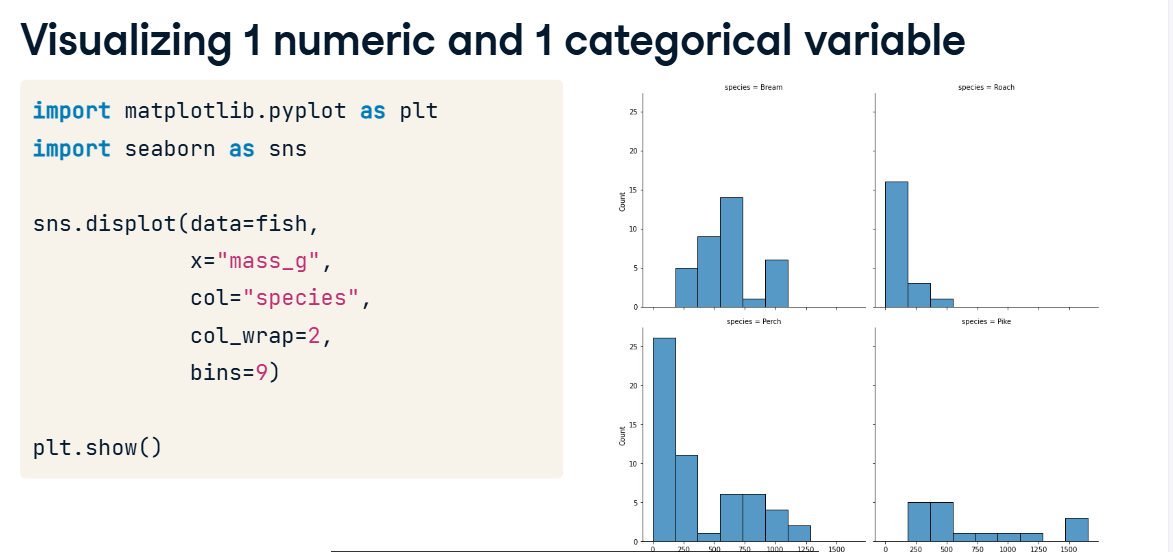
Let's take a look at some data on the masses of fish sold at a fish market. Each row of data contains the species of a fish, and its mass. The mass will be the response variable.



**3. Visualizing 1 numeric and 1 categorical variable**

00:23 - 01:04

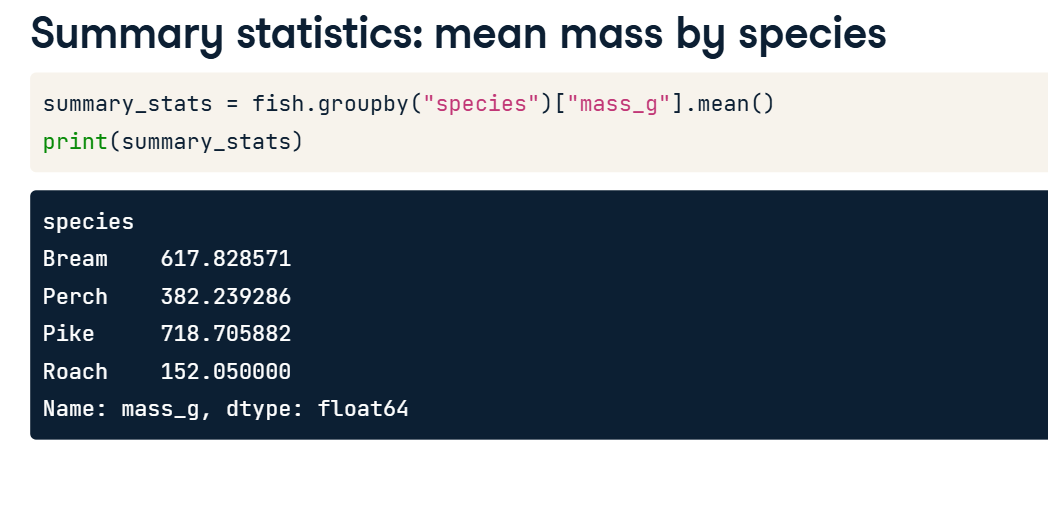
To visualize the data, scatter plots aren't ideal because species is categorical. Instead, we can draw a histogram for each of the species. To give a separate panel to each species, I use seaborn's displot function. This takes a DataFrame as the data argument, the variable of interest as x, and the variable you want to split on as col. It also takes an optional col\_wrap argument to specify the number of plots per row. Because the dataset is fairly small, I also set the bins argument to nine. By default, displot creates histograms.



**4. Summary statistics: mean mass by species**

01:04 - 01:23

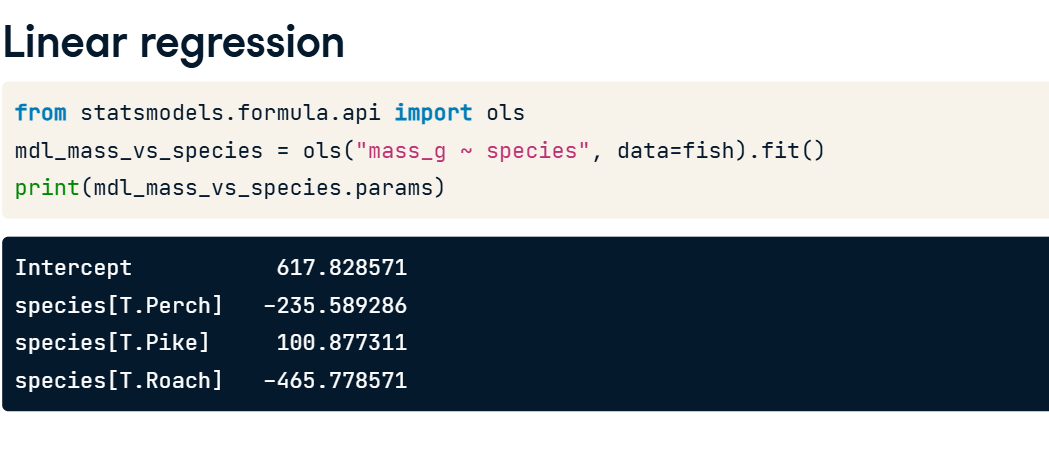
Let's calculate some summary statistics. First we group by species, then we calculate their mean masses. You can see that the mean mass of a bream is six hundred and eighteen grams. The mean mass for a perch is three hundred and eighty two grams, and so on.



**5. Linear regression**

01:23 - 02:14

Let's run a linear regression using mass as the response variable and species as the explanatory variable. The syntax is the same: you call ols(), passing a formula with the response variable on the left and the explanatory variable on the right, and setting the data argument to the DataFrame. We fit the model using the fit method, and retrieve the parameters using dot params on the fitted model. This time we have four coefficients: an intercept, and one for three of the fish species. A coefficient for bream is missing, but the number for the intercept looks familiar. The intercept is the mean mass of the bream that you just calculated. You might wonder what the other coefficients are, and why perch has a negative coefficient, since fish masses can't be negative.



**6. Model with or without an intercept**

02:14 - 03:18

The coefficients for each category are calculated relative to the intercept. This way of displaying results can be useful for models with multiple explanatory variables, but for simple linear regression, it's just confusing. Fortunately, we can fix it. By changing the formula slightly to append "plus zero", we specify that all the coefficients should be given relative to zero. Equivalently, it means we are fitting a linear regression without an intercept term. If you subtract two hundred and thirty five point fifty-nine from six hundred and seventeen point eighty-three, you get three hundred and eighty two point twenty four, which is the mean mass of a perch. Now these coefficients make more sense. They are all just the mean masses for each species. This is a reassuringly boring result. When you only have a single, categorical explanatory variable, the linear regression coefficients are simply the means of each category.

