Linear Regression

By Evgenia "Jenny" Nitishinskaya and Delaney Granizo-Mackenzie with example algorithms by David Edwards

Part of the Quantopian Lecture Series:

- www.quantopian.com/lectures (https://www.quantopian.com/lectures)
- github.com/quantopian/research_public (https://github.com/quantopian/research_public)

Linear regression is a technique that measures the relationship between two variables. If we have an independent variable X, and a dependent outcome variable Y, linear regression allows us to determine which linear model $Y = \alpha + \beta X$ best explains the data. As an example, let's consider TSLA and SPY. We would like to know how TSLA varies as a function of how SPY varies, so we will take the daily returns of each and regress them against each other.

Python's `statsmodels` library has a built-in linear fit function. Note that this will give a line of best fit; whether or not the relationship it shows is significant is for you to determine. The output will also have some statistics about the model, such as R-squared and the F value, which may help you quantify how good the fit actually is.

In [1]:
```
# Import libraries
import numpy as np
from statsmodels import regression
import statsmodels.api as sm
import matplotlib.pyplot as plt
import math
```

First we'll define a function that performs linear regression and plots the results.

In [2]:
```
def linreg(X,Y):
    # Running the linear regression
    X = sm.add_constant(X)
    model = regression.linear_model.OLS(Y, X).fit()
    a = model.params[0]
    b = model.params[1]
    X = X[:, 1]

    # Return summary of the regression and plot results
    X2 = np.linspace(X.min(), X.max(), 100)
    Y_hat = X2 * b + a
    plt.scatter(X, Y, alpha=0.3) # Plot the raw data
    plt.plot(X2, Y_hat, 'r', alpha=0.9);  # Add the regression line, colored in red
    plt.xlabel('X Value')
    plt.ylabel('Y Value')
    return model.summary()
```

Now we'll get pricing data on TSLA and SPY and perform a regression.

In [3]:
```
start = '2014-01-01'
end = '2015-01-01'
asset = get_pricing('TSLA', fields='price', start_date=start, end_date=end)
benchmark = get_pricing('SPY', fields='price', start_date=start, end_date=end)

# We have to take the percent changes to get to returns
# Get rid of the first (0th) element because it is NAN
r_a = asset.pct_change()[1:]
r_b = benchmark.pct_change()[1:]

linreg(r_b.values, r_a.values)
```

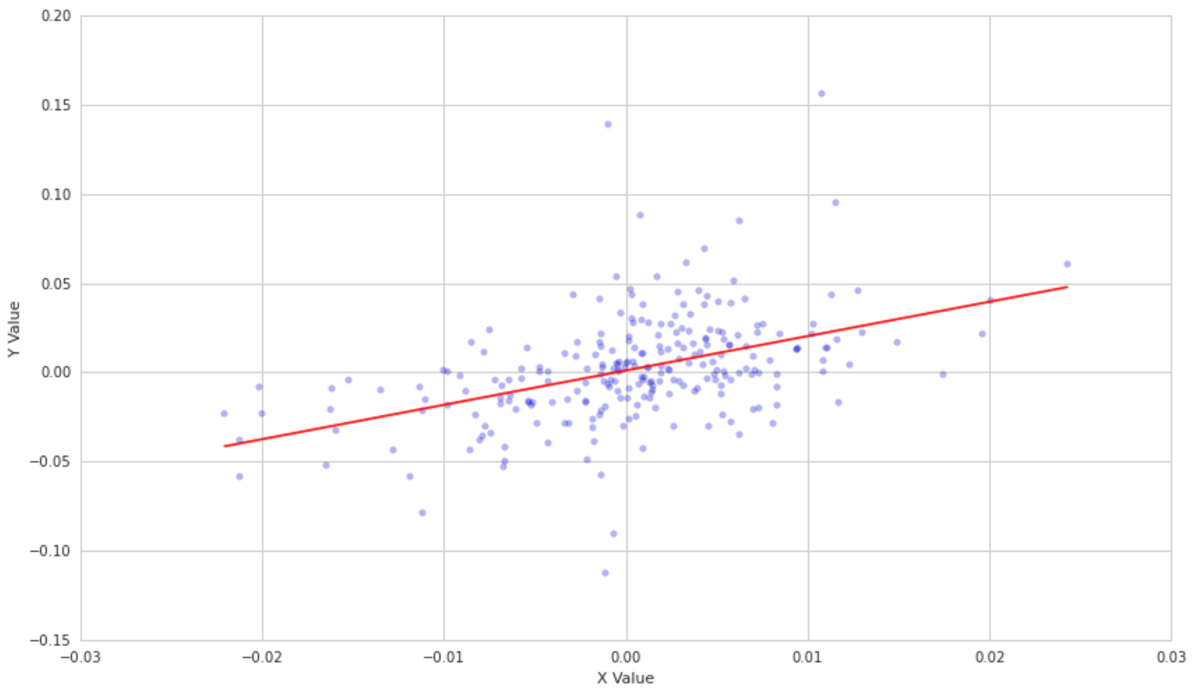
Out[3]:

Dep. Variable:	y	**R-squared:**	0.202
Model:	OLS	**Adj. R-squared:**	0.199
Method:	Least Squares	**F-statistic:**	63.14
Date:	Thu, 17 Sep 2015	**Prob (F-statistic):**	6.66e-14
Time:	20:55:12	**Log-Likelihood:**	548.81
No. Observations:	251	**AIC:**	-1094.

Df Residuals:	249	**BIC:**	-1087.
Df Model:	1		
Covariance Type:	nonrobust		

	coef	std err	t	P>|t|	[95.0% Conf. Int.]
const	0.0011	0.002	0.626	0.532	-0.002 0.004
x1	1.9271	0.243	7.946	0.000	1.449 2.405

Omnibus:	63.947	**Durbin-Watson:**	2.006
Prob(Omnibus):	0.000	**Jarque-Bera (JB):**	371.184
Skew:	0.850	**Prob(JB):**	2.50e-81
Kurtosis:	8.710	**Cond. No.**	141.



Each point on the above graph represents a day, with the x-coordinate being the return of SPY, and the y-coordinate being the return of TSLA. As we can see, the line of best fit tells us that for every 1% increased return we see from the SPY, we should see an extra 1.92% from TSLA. This is expressed by the parameter β, which is 1.9271 as estimated. Of course, for decresed return we will also see about double the loss in TSLA, so we haven't gained anything, we are just more volatile.

Linear Regression vs. Correlation

- Linear regression gives us a specific linear model, but is limited to cases of linear dependence.
- Correlation is general to linear and non-linear dependencies, but doesn't give us an actual model.
- Both are measures of covariance.
- Linear regression can give us relationship between Y and many independent variables by making X multidimensional.

Knowing Parameters vs. Estimates

It is very important to keep in mind that all α and β parameters estimated by linear regression are just that - estimates. You can never know the underlying true parameters unless you know the physical process producing the data. The paremeters you estimate today may not be the same as the same analysis done including tomorrow's data, and the underlying true parameters may be moving. As such it is very important when doing actual analysis to pay attention to the standard error of the parameter estimates. More material on the standard error will be presented in a later lecture. One way to get a sense of how stable your paremeter estimates are is to estimates them using a rolling window of data and see how much variance there is in the estimates.

Example case

Now let's see what happens if we regress two purely random variables.

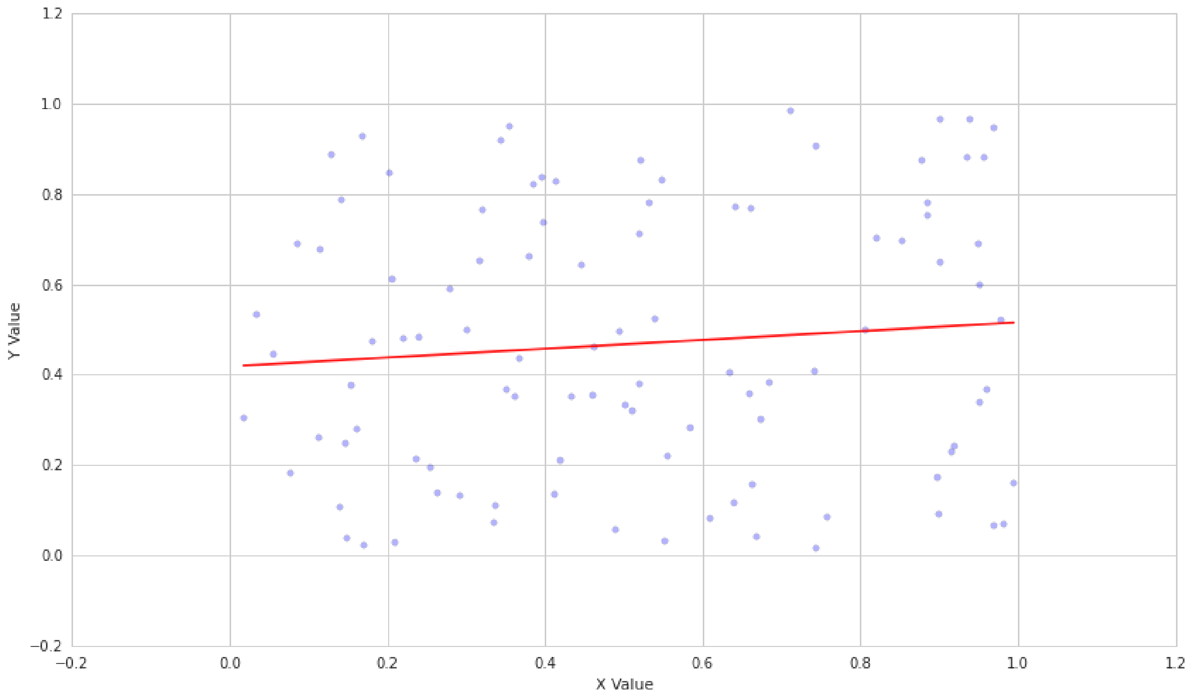
In [4]:
```
X = np.random.rand(100)
Y = np.random.rand(100)
linreg(X, Y)
```

Out[4]:

Dep. Variable:	y	**R-squared:**	0.009
Model:	OLS	**Adj. R-squared:**	-0.001
Method:	Least Squares	**F-statistic:**	0.9024
Date:	Thu, 17 Sep 2015	**Prob (F-statistic):**	0.344
Time:	20:55:13	**Log-Likelihood:**	-19.046
No. Observations:	100	**AIC:**	42.09
Df Residuals:	98	**BIC:**	47.30
Df Model:	1		
Covariance Type:	nonrobust		

	coef	std err	t	P>|t|	[95.0% Conf. Int.]
const	0.4187	0.061	6.914	0.000	0.299 0.539
x1	0.0972	0.102	0.950	0.344	-0.106 0.300

Omnibus:	48.699	**Durbin-Watson:**	2.054
Prob(Omnibus):	0.000	**Jarque-Bera (JB):**	6.924
Skew:	0.091	**Prob(JB):**	0.0314
Kurtosis:	1.724	**Cond. No.**	4.45



The above shows a fairly uniform cloud of points. It is important to note that even with 100 samples, the line has a visible slope due to random chance. This is why it is crucial that you use statistical tests and not visualizations to verify your results.

Now let's make Y dependent on X plus some random noise.

In [5]:
```
# Generate ys correlated with xs by adding normally-destributed errors
Y = X + 0.2*np.random.randn(100)

linreg(X,Y)
```

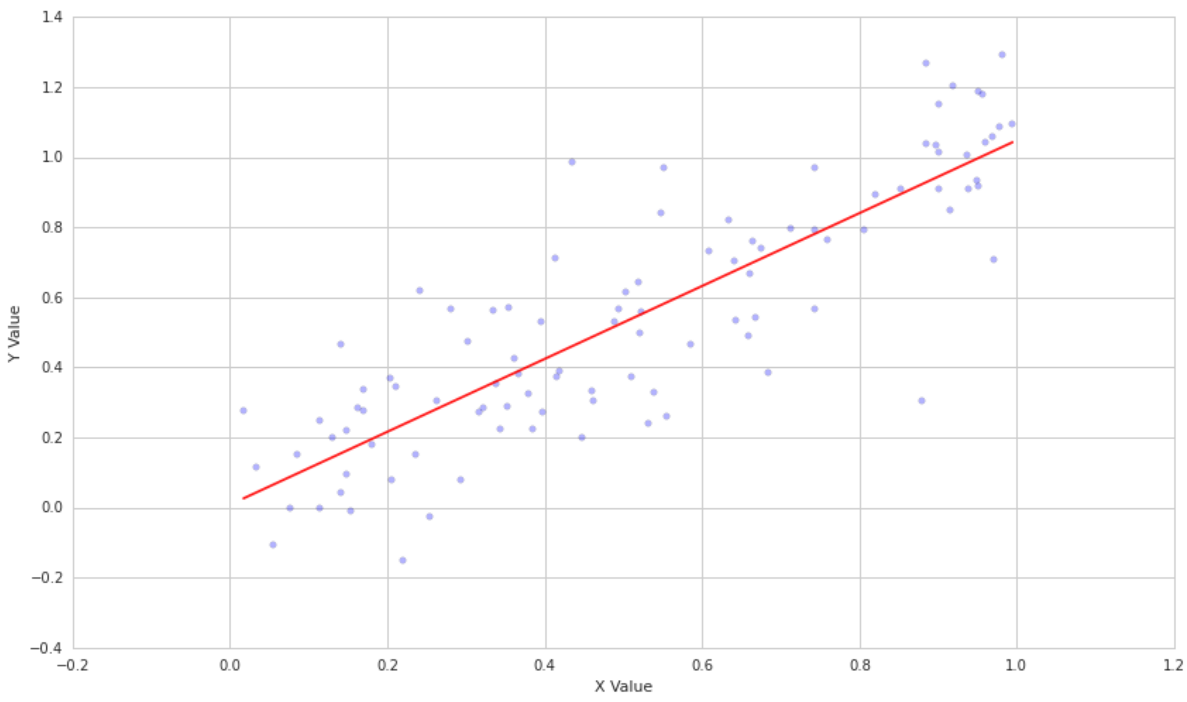
Out[5]:

Dep. Variable:	y	**R-squared:**	0.727

Model:	OLS	**Adj. R-squared:**	0.724
Method:	Least Squares	**F-statistic:**	261.2
Date:	Thu, 17 Sep 2015	**Prob (F-statistic):**	2.14e-29
Time:	20:55:13	**Log-Likelihood:**	27.312
No. Observations:	100	**AIC:**	-50.62
Df Residuals:	98	**BIC:**	-45.41
Df Model:	1		
Covariance Type:	nonrobust		

	coef	std err	t	P>|t|	[95.0% Conf. Int.]
const	0.0081	0.038	0.213	0.832	-0.067 0.084
x1	1.0405	0.064	16.162	0.000	0.913 1.168

Omnibus:	2.687	**Durbin-Watson:**	1.645
Prob(Omnibus):	0.261	**Jarque-Bera (JB):**	2.341
Skew:	-0.138	**Prob(JB):**	0.310
Kurtosis:	3.697	**Cond. No.**	4.45



In a situation like the above, the line of best fit does indeed model the dependent variable Y quite well (with a high R^2 value).

Evaluating and reporting results

The regression model relies on several assumptions:

- The independent variable is not random.
- The variance of the error term is constant across observations. This is important for evaluating the goodness of the fit.
- The errors are not autocorrelated. The Durbin-Watson statistic detects this; if it is close to 2, there is no autocorrelation.
- The errors are normally distributed. If this does not hold, we cannot use some of the statistics, such as the F-test.

If we confirm that the necessary assumptions of the regression model are satisfied, we can safely use the statistics reported to analyze the fit. For example, the R^2 value tells us the fraction of the total variation of Y that is explained by the model.

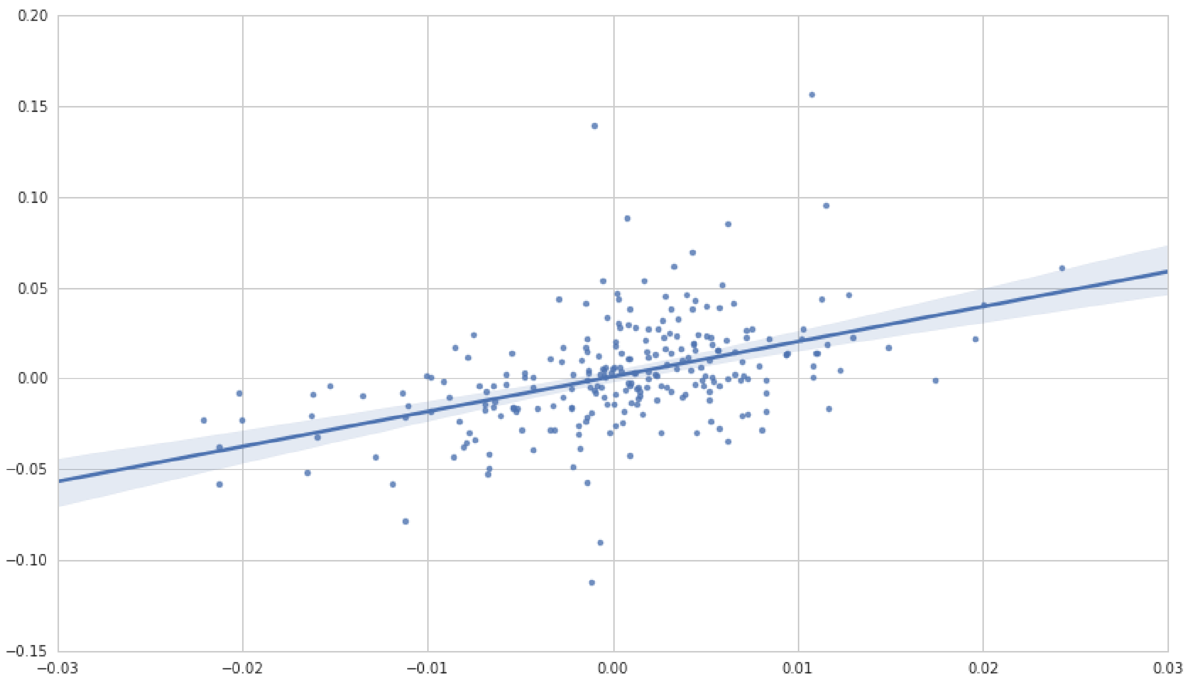
When making a prediction based on the model, it's useful to report not only a single value but a confidence interval. The linear regression reports 95% confidence intervals for the regression parameters, and we can visualize what this means using the `seaborn` library, which plots the regression line and highlights the 95% (by default) confidence interval for the regression line:

In [6]:
```
import seaborn

start = '2014-01-01'
end = '2015-01-01'
asset = get_pricing('TSLA', fields='price', start_date=start, end_date=end)
benchmark = get_pricing('SPY', fields='price', start_date=start, end_date=end)

# We have to take the percent changes to get to returns
# Get rid of the first (0th) element because it is NAN
r_a = asset.pct_change()[1:]
r_b = benchmark.pct_change()[1:]

seaborn.regplot(r_b.values, r_a.values);
```



Mathematical Background

This is a very brief overview of linear regression. For more, please see: https://en.wikipedia.org/wiki/Linear_regression (https://en.wikipedia.org/wiki/Linear_regression)

Ordinary Least Squares

Regression works by optimizing the placement of the line of best fit (or plane in higher dimensions). It does so by defining how bad the fit is using an objective function. In ordinary least squares regression (OLS), what we use here, the objective function is:

$$\sum_{i=1}^n (Y_i - a - bX_i)^2$$

We use a and b to represent the potential candidates for α and β. What this objective function means is that for each point on the line of best fit we compare it with the real point and take the square of the difference. This function will decrease as we get better parameter estimates. Regression is a simple case of numerical optimization that has a closed form solution and does not need any optimizer. We just find the results that minimize the objective function.

We will denote the eventual model that results from minimizing our objective function as:

$$\hat{Y} = \hat{\alpha} + \hat{\beta}X$$

With $\hat{\alpha}$ and $\hat{\beta}$ being the chosen estimates for the parameters that we use for prediction and $\hat{Y}$ being the predicted values of Y given the estimates.

Standard Error

We can also find the standard error of estimate, which measures the standard deviation of the error term ϵ, by getting the `scale` parameter of the model returned by the regression and taking its square root. The formula for standard error of estimate is

$$s = \left(\frac{\sum_{i=1} \epsilon_i}{n-2} \right)$$

If $\hat{\alpha}$ and $\hat{\beta}$ were the true parameters ($\hat{\alpha} = \alpha$ and $\hat{\beta} = \beta$), we could represent the error for a particular predicted value of Y as s^2 for all values of X_i. We could simply square the difference $(Y - \hat{Y})$ to get the variance because $\hat{Y}$ incorporates no error in the paremeter estimates themselves. Because $\hat{\alpha}$ and $\hat{\beta}$ are merely estimates in our construction of the model of Y, any predicted values , $\hat{Y}$, will have their own standard error based on the distribution of the X terms that we plug into the model. This forecast error is represented by the following:

$$s_f^2 = s^2\left(1 + \frac{1}{n} + \frac{(X - \mu_X)^2}{(n-1)\sigma_X^2}\right)$$

where μ_X is the mean of our observations of X and σ_X is the standard deviation of X. This adjustment to s^2 incorporates the uncertainty in our parameter estimates. Then the 95% confidence interval for the prediction is $\hat{Y} \pm t_c s_f$, where t_c is the critical value of the t-statistic for n samples and a desired 95% confidence.

This presentation is for informational purposes only and does not constitute an offer to sell, a solicitation to buy, or a recommendation for any