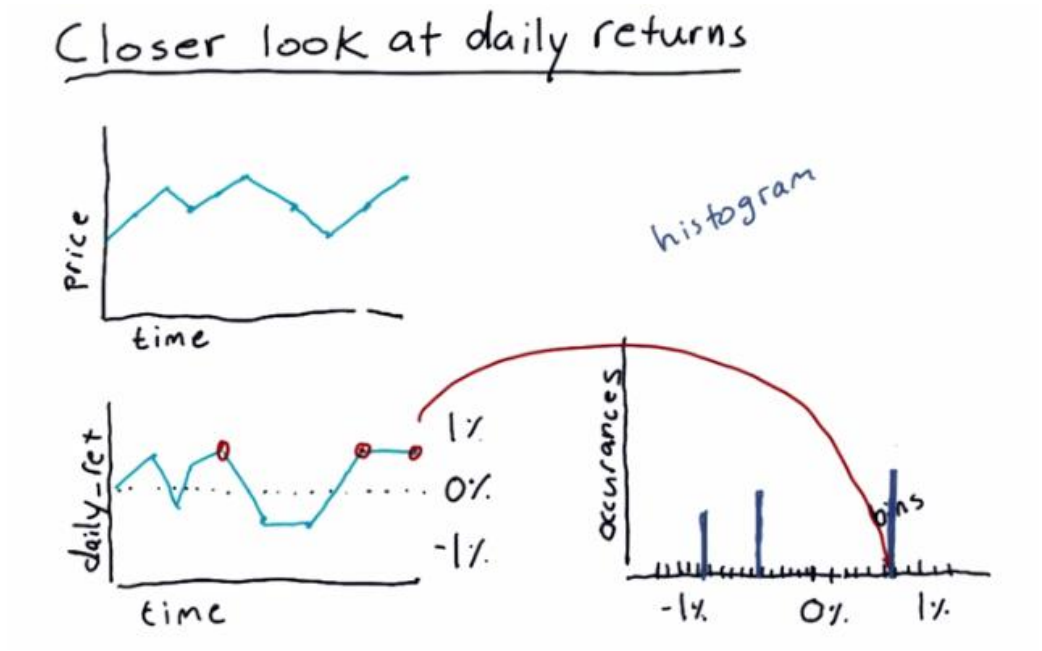
01-06 Histograms and Scatter Plots

Compiled by Shipra De, Fall 2016

Histograms and Scatter Plots

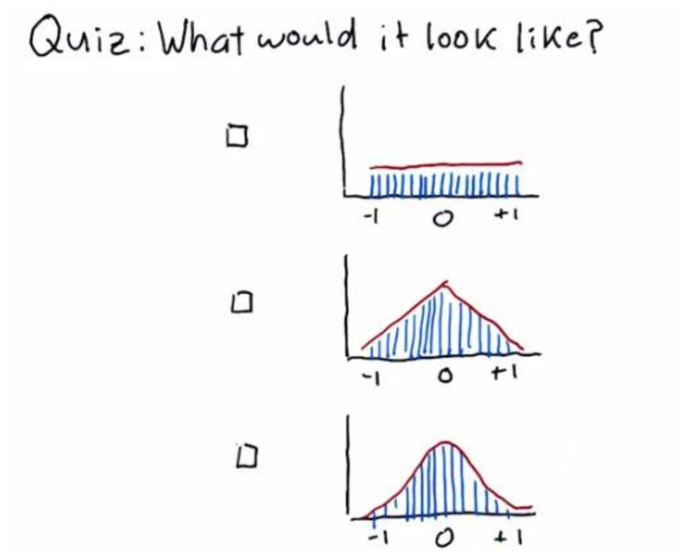
- Daily returns are one of the most important factors for us to consider when looking at market statistics. But daily returns for a single stock just by themselves are not very informative.
- One of the most informative ways to consider daily returns is when we compare the returns of one stock with another.

A Closer Look at Daily Returns

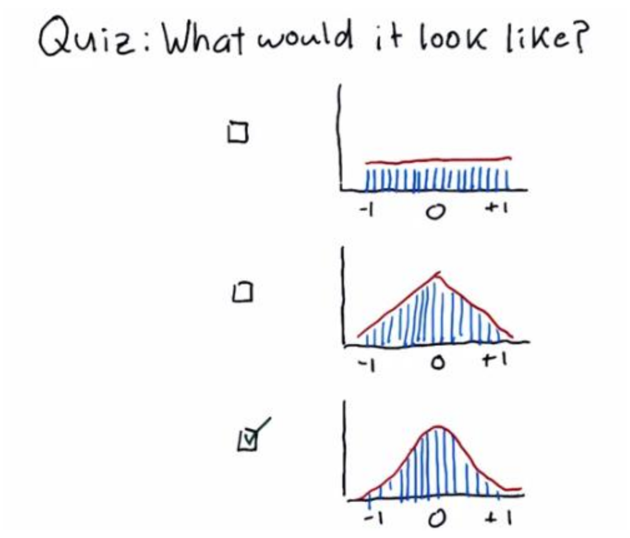


- We're going to use daily returns as a basis for the analysis in this lesson and we build daily returns, like we've seen before, by starting with a price time series.
- And each point in this daily return chart is related to how much price has changed on that corresponding day.
- So, for instance, this represents how much the price changed from this day to that day, about 1%. And, of course, we have that for each of the days in our history.
- Now looking at this data, this daily return data, it's not too revealing. It's hard to draw any sorts of interesting conclusions just by visually looking at this data from day to day.
- And so, there's a number of interesting ways that we can look at that data, and that's what this lesson is all about.
- Those two ways are histograms and scatter plots. Let's start by taking a look at histograms.
- A histogram is a kind of bar chart where we plot the number of occurrences of each item versus the value. So the way we accomplish that is, we split up the range of data into lots of little bins. And we count up how many times the data matches the range across that bin.
- So, as an example, if you notice here we've got several occurrences of this value, which is about the same, and those three occurrences are probably in say, this bin. So when we go to plot the histogram overall, we would see a bar of the appropriate height here that represents how many times the data matched that value.
- And of course, we have values in other bins, and so the bars in those bins would have various heights. And as you gather that data across all of time, a shape emerges of this histogram, and that provides a lot of information.
- So let's consider what that shape might look like.

What Would It Look Like?

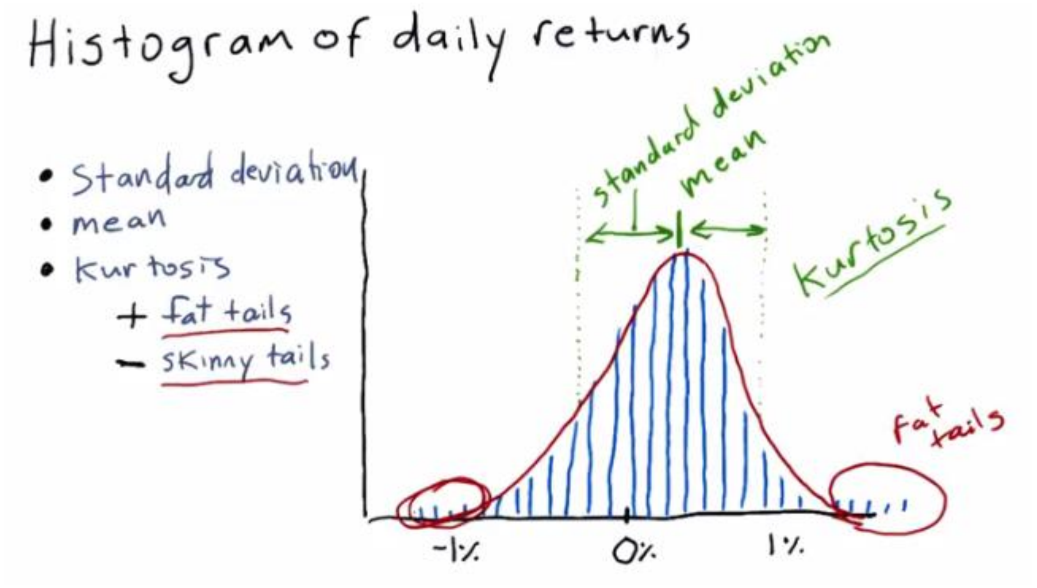


- Suppose now that we've looked at, say the S&P 500 over many years and we've measured each day what the daily return is for the S&P 500. And we conduct a histogram and plot that histogram. So, for each little bin we have a bar there.
- What is the shape of this histogram going to look like? Do you think the bars, when we put them all together, will have a sort of flat shape, maybe a triangular shape, or something that's more like a bell curve?
- Check the box next to the histogram that you think is the best answer.



- The correct answer is bell curve. [COUGH] That's what many, many distributions in nature.
- And if you consider, [LAUGH] the stock market nature, is not unusual that a histogram of daily returns ends up looking like a bell curve.

Histogram of Daily Returns



- Once we've got our histogram, there are a lot of statistics we can run on it to characterize it.
- For instance, of course we might be interested in the mean. We might also be interested in the standard deviation. Which is essentially on average how far do individual measurements deviate from the mean.
- Another very important measure is something called Kurtosis. Kurtosis comes from a Greek word that means curved or arching.
- So what does Kurtosis mean? Well let me show you. Kurtosis tells us about the tails of the distribution. So the tails are the parts out here towards the ends.
- And if we assume that our distribution is similar to a Gaussian distribution, or normal distribution. The measure of kurtosis tells us how much different our histogram is from that traditional Gaussian distribution. So in this case we have what are called fat tails. We got them over here and over here.
- What that means is that there are occasional, and more frequent than would happen if we had a regular Gaussian distribution. There are frequently large excursions more frequently than if this was a normal distribution.
- If you were to measure the kurtosis of this histogram, you would get a positive number. Meaning that there are more occurrences out in these tails than would be expected if it were a normal distribution.
- If you measured a negative kurtosis it would mean that there are many fewer occurrences out here on the tails than would be expected if it were a normal distribution.
- So we can plot our data in this sort of bar chart called a histogram. We can measure statistics on it like standard deviation, mean and kurtosis.
- And remember the following about kurtosis. If we’ve got a positive kurtosis, that means we’ve got fat tails, like in this example. There’s more occurrences outside in the tails than would

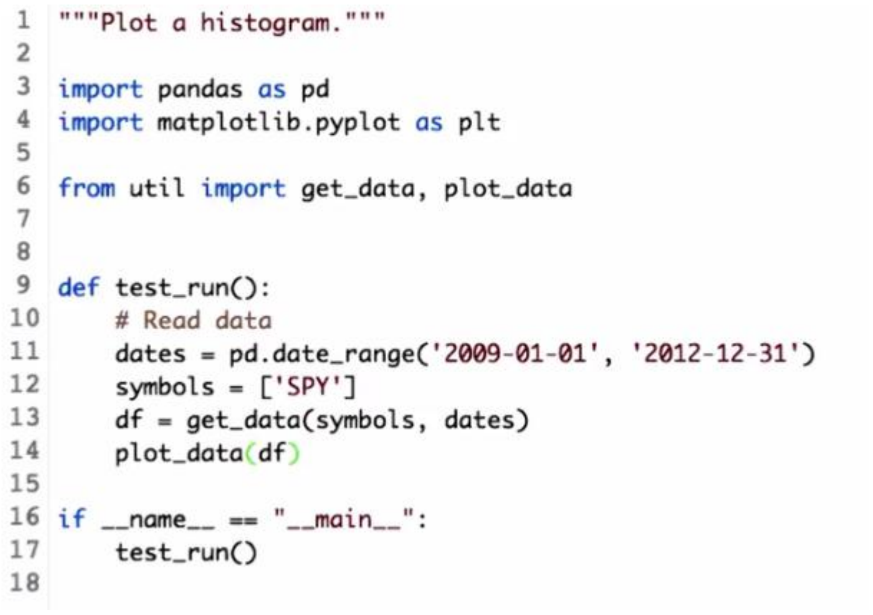
normally happen with a Gaussian distribution. And if we’ve got a negative kurtosis, we’ve got skinny tails, meaning there’s less out there.

- Now, I’m going to hand it over to Dave, and she’s going to show you how to make this plot and calculate these numbers in Python.

How To Plot a Histogram

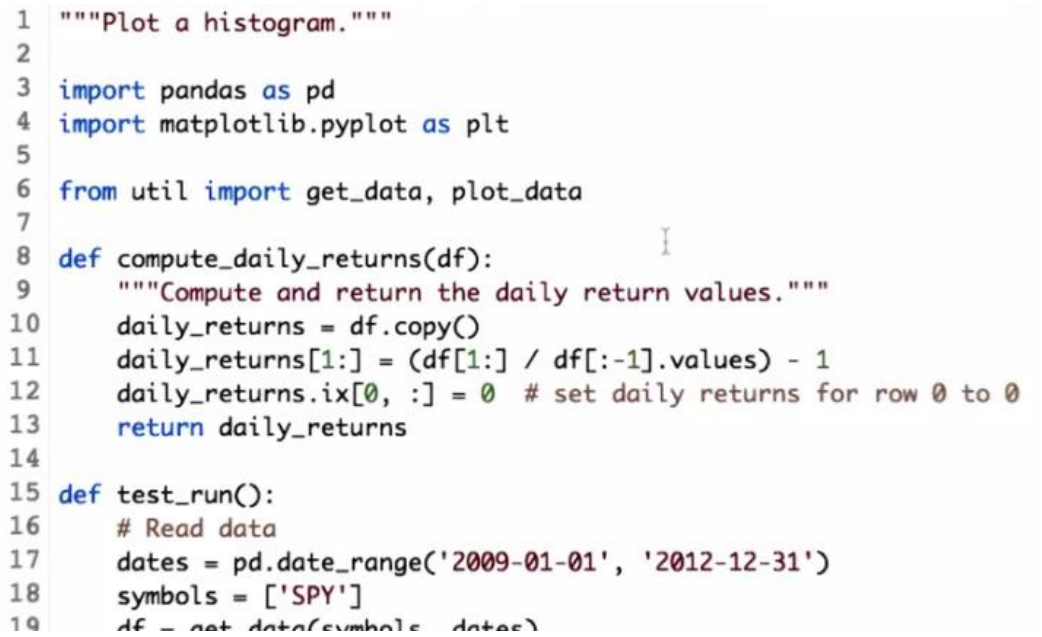
"I'm attending a programmers' retreat right now. I'm working on an app that will hopefully provide some useful visualizations of how British politicians vote on certain bills. I feel like programming is the closest thing we have to magic. It allows you to create things with words."

- I recently read a post on Humans of New York page, and this woman mentioned in her interview that programming is magic. It allows us to make things with words. Isn't that true?
- So let's create our own magic. Let's make histogram. Firstly, let's check out the ingredients needed to make histogram of daily returns.



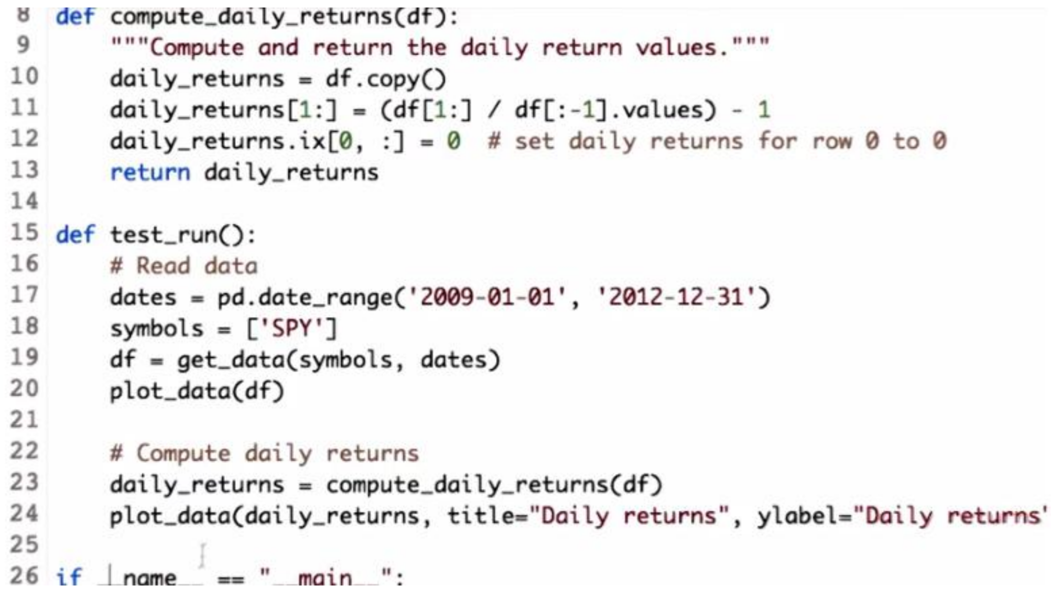
```python
"""Plot a histogram."""

import pandas as pd
import matplotlib.pyplot as plt

from util import get_data, plot_data


def test_run():
    # Read data
    dates = pd.date_range('2009-01-01', '2012-12-31')
    symbols = ['SPY']
    df = get_data(symbols, dates)
    plot_data(df)

if __name__ == "__main__":
    test_run()

```

- To calculate daily returns, we need to get stock prices first. To start with, we get stock prices for the SPY for a period of three years. Next step is to calculate daily return.



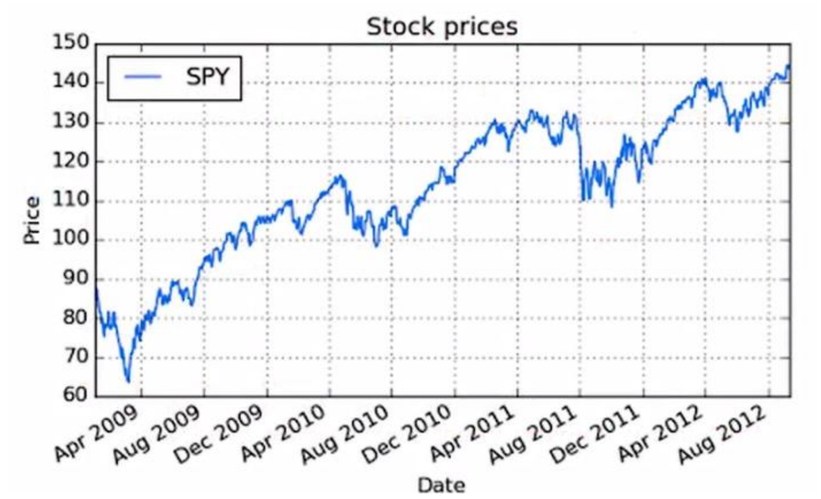
```
"""Plot a histogram."""

import pandas as pd
import matplotlib.pyplot as plt

from util import get_data, plot_data

def compute_daily_returns(df):
    """Compute and return the daily return values."""
    daily_returns = df.copy()
    daily_returns[1:] = (df[1:] / df[:-1].values) - 1
    daily_returns.ix[0, :] = 0  # set daily returns for row 0 to 0
    return daily_returns

def test_run():
    # Read data
    dates = pd.date_range('2009-01-01', '2012-12-31')
    symbols = ['SPY']
    df = get_data(symbols, dates)
```

- This is similar to what we saw in lesson four. Now we call this function and pass a DataFrame to it.

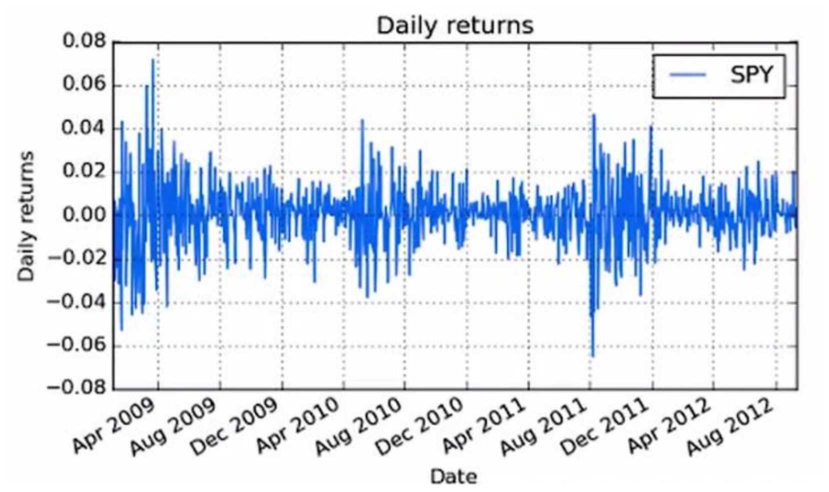


```
def compute_daily_returns(df):
    """Compute and return the daily return values."""
    daily_returns = df.copy()
    daily_returns[1:] = (df[1:] / df[:-1].values) - 1
    daily_returns.ix[0, :] = 0  # set daily returns for row 0 to 0
    return daily_returns

def test_run():
    # Read data
    dates = pd.date_range('2009-01-01', '2012-12-31')
    symbols = ['SPY']
    df = get_data(symbols, dates)
    plot_data(df)

    # Compute daily returns
    daily_returns = compute_daily_returns(df)
    plot_data(daily_returns, title="Daily returns", ylabel="Daily returns'

if __name__ == "__main__":
```

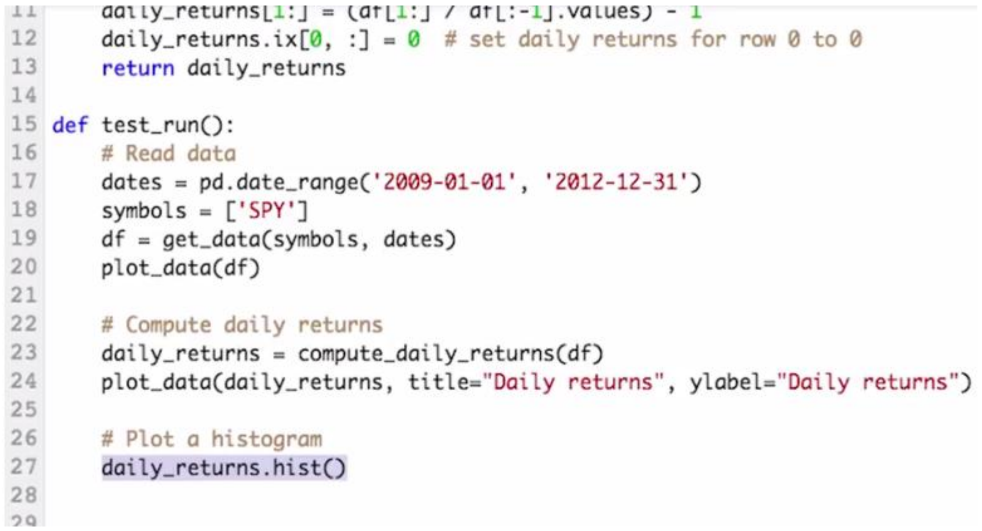
- We also plot the daily return value by passing the daily return DataFrame to our plot data_function.



- This graph show the prices of the SPY stock over three years.

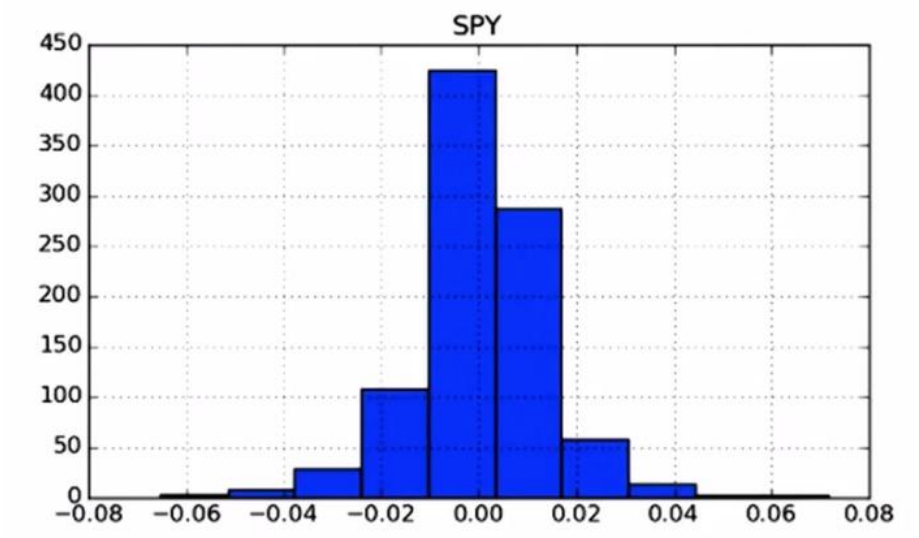


- And this is the daily return graph for SPY. Now we have our base ready, so let's make our histogram.

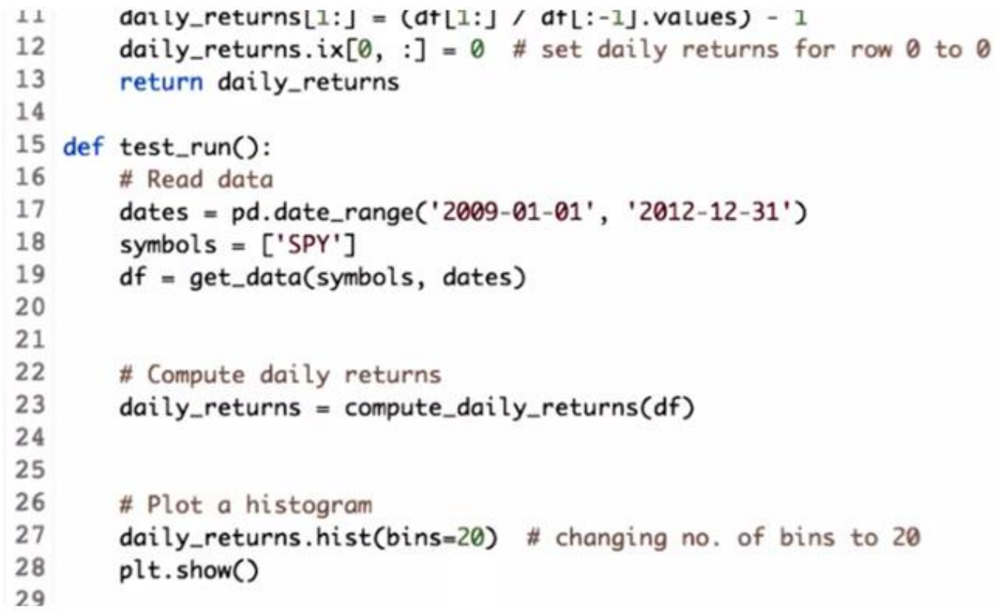


```
    daily_returns[1:] = (df[1:] / df[:-1].values) - 1
    daily_returns.ix[0, :] = 0  # set daily returns for row 0 to 0
    return daily_returns

def test_run():
    # Read data
    dates = pd.date_range('2009-01-01', '2012-12-31')
    symbols = ['SPY']
    df = get_data(symbols, dates)
    plot_data(df)

    # Compute daily returns
    daily_returns = compute_daily_returns(df)
    plot_data(daily_returns, title="Daily returns", ylabel="Daily returns")

    # Plot a histogram
    daily_returns.hist()

```

- And after five lessons, you must have guessed that even this can be done in just one line.

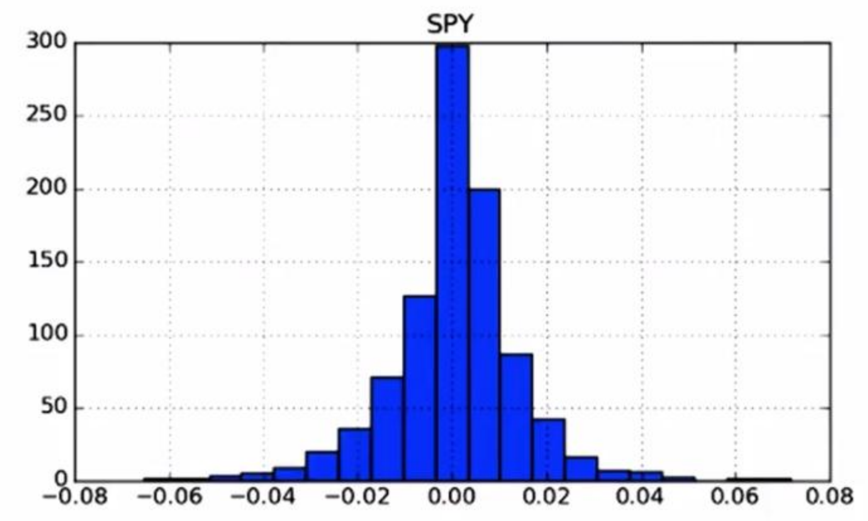


- Here is our histogram. Professor explained the concept about bins. We did not mention the number of bins while plotting the histogram. The default number of bins is 10.
- If you look closely, you will observe 10 sets of ranges. Can you see this? 1, 2, 3, 4 and so on. If you count, there would be ten.
- But as usual, Python is flexible and allows us to change the number of bins, using the bin keyword.



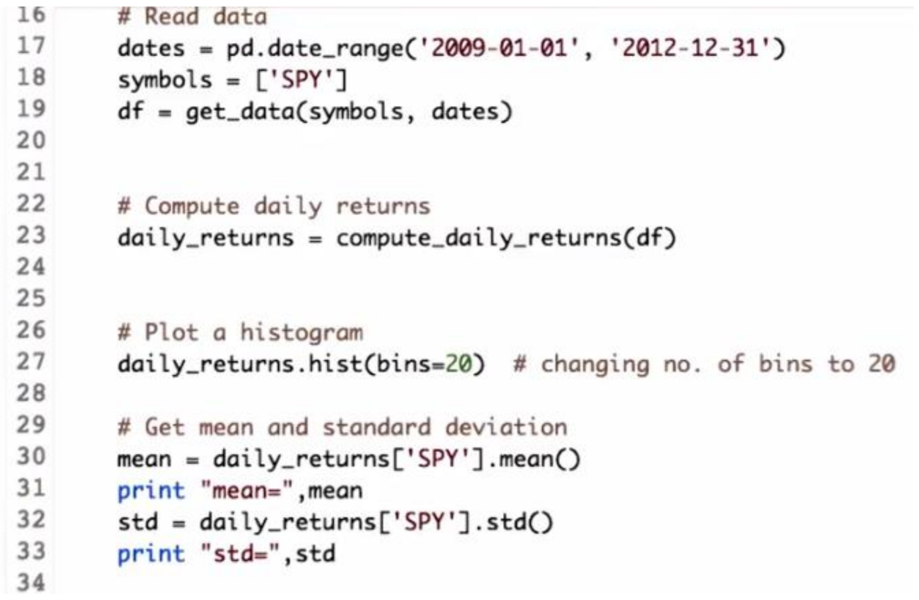
```
    daily_returns[1:] = (df[1:] / df[:-1].values) - 1
    daily_returns.ix[0, :] = 0  # set daily returns for row 0 to 0
    return daily_returns

def test_run():
    # Read data
    dates = pd.date_range('2009-01-01', '2012-12-31')
    symbols = ['SPY']
    df = get_data(symbols, dates)


    # Compute daily returns
    daily_returns = compute_daily_returns(df)


    # Plot a histogram
    daily_returns.hist(bins=20)  # changing no. of bins to 20
    plt.show()

```

- We inform the histogram function that we need to empty bins by passing a parameter bins and assigning it a value 20. Now let's check our changed histogram.



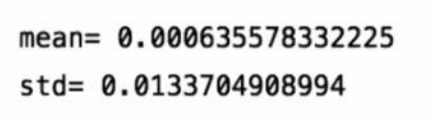
- Notice that the width of each bar has reduced. And the number of bars has increased. If you read this graph, it would say that there are approximately 300 values which lie near 0.
- Next we compute some statistics on the daily return.

Computing Histogram Statistics



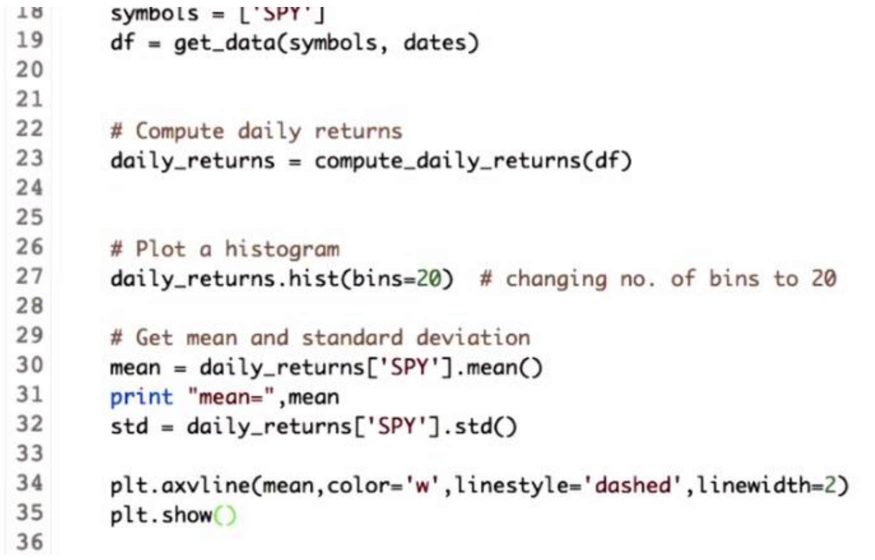
```
    # Read data
    dates = pd.date_range('2009-01-01', '2012-12-31')
    symbols = ['SPY']
    df = get_data(symbols, dates)


    # Compute daily returns
    daily_returns = compute_daily_returns(df)


    # Plot a histogram
    daily_returns.hist(bins=20)  # changing no. of bins to 20

    # Get mean and standard deviation
    mean = daily_returns['SPY'].mean()
    print "mean=",mean
    std = daily_returns['SPY'].std()
    print "std=",std

```

- Starting with mean and standard deviation. We call the function mean and std on our dataframe to get the values. Let's go ahead and check the mean and the standard deviation for the daily returns of SPY stock.



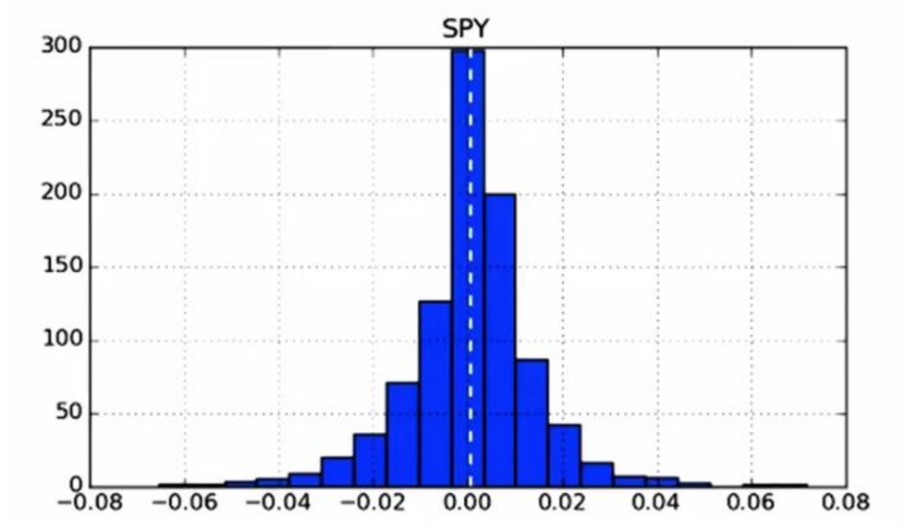
```
mean= 0.000635578332225
std= 0.0133704908994
```

- So we get the mean and the standard deviation.
- Happy with just knowing the mean and standard deviation value? But I am not. I want to see it on the plot, just like Professor did. So let's learn how to add mean and standard deviation line on plot.

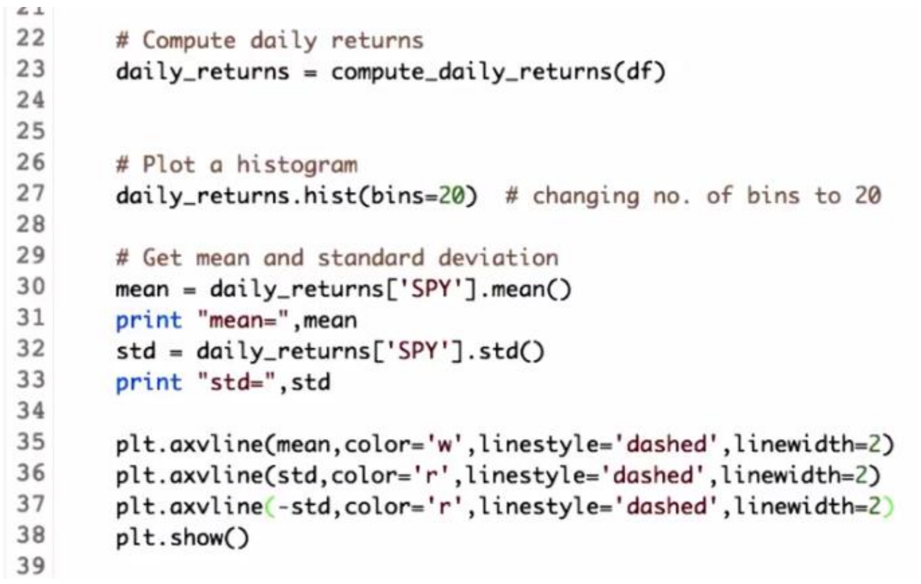


```
    symbols = ['SPY']
    df = get_data(symbols, dates)


    # Compute daily returns
    daily_returns = compute_daily_returns(df)


    # Plot a histogram
    daily_returns.hist(bins=20)  # changing no. of bins to 20

    # Get mean and standard deviation
    mean = daily_returns['SPY'].mean()
    print "mean=",mean
    std = daily_returns['SPY'].std()

    plt.axvline(mean,color='w',linestyle='dashed',linewidth=2)
    plt.show()

```

- Matplotlib library has a function axvline. Looking at the substring vline, we can guess it will give vertical line.
- Let's check out its parameters. First we pass the mean value, then just for beautification and so that we can differentiate the mean line from the rest histogram, we add a color, which is white, make it a dashed style line and increase the linewidth to two.
- Now let's check our output.

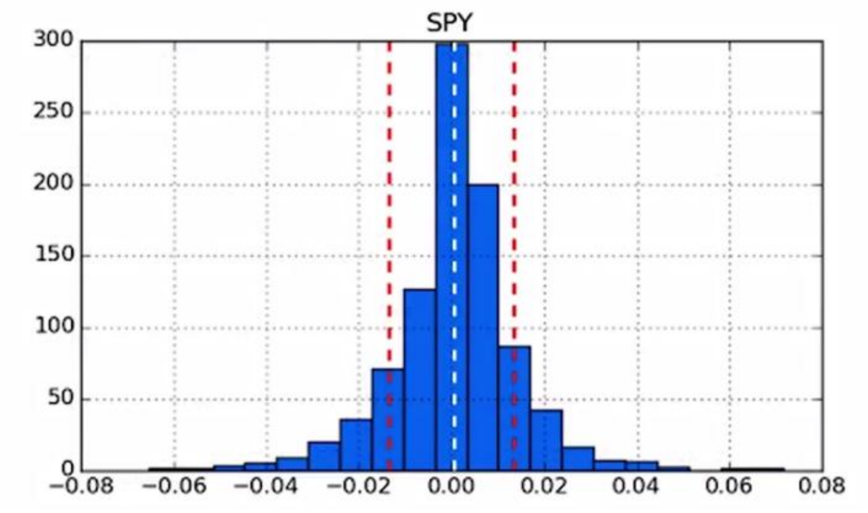


- So this our mean and this is how it is plotted on the histogram.

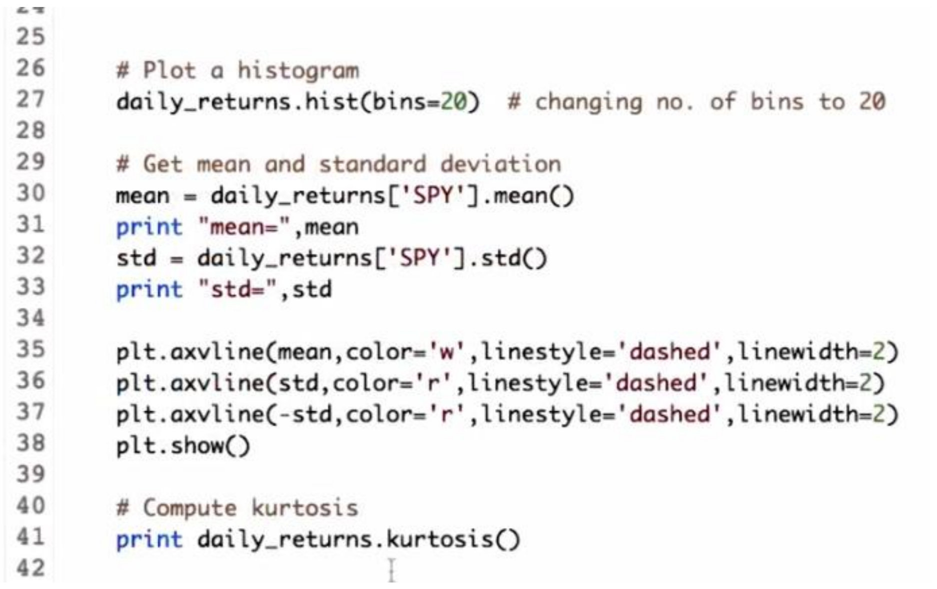


```
    # Compute daily returns
    daily_returns = compute_daily_returns(df)


    # Plot a histogram
    daily_returns.hist(bins=20)  # changing no. of bins to 20

    # Get mean and standard deviation
    mean = daily_returns['SPY'].mean()
    print "mean=",mean
    std = daily_returns['SPY'].std()
    print "std=",std

    plt.axvline(mean,color='w',linestyle='dashed',linewidth=2)
    plt.axvline(std,color='r',linestyle='dashed',linewidth=2)
    plt.axvline(-std,color='r',linestyle='dashed',linewidth=2)
    plt.show()

```

- Now let's go ahead and plot standard deviation. To plot the standard deviation line, it is similar to the mean.
- But as we want the standard deviation line on both side of the mean, we plot it twice. One with the positive value and one with the negative value, to show standard deviation line on either side of the mean.

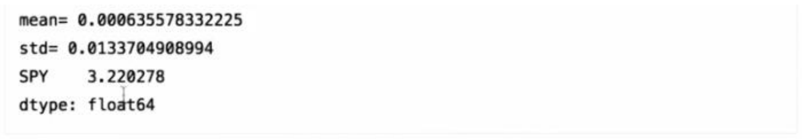


- Ta dah, we have our standard deviation lines on our graph. To give the standard deviation a red color, I have just replaced the parameter color of white, with red. Now I'm happy with the graph and I hope you do are.
- Let's move ahead to kurtosis. We can expect that dataframe would have a function for calculating kurtosis as well.

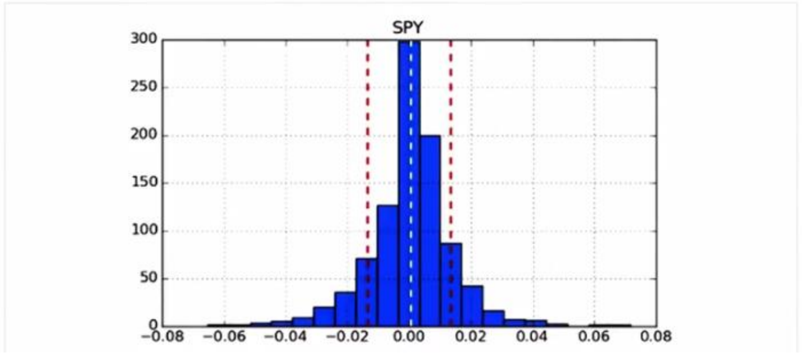


```
# Plot a histogram
daily_returns.hist(bins=20)  # changing no. of bins to 20

# Get mean and standard deviation
mean = daily_returns['SPY'].mean()
print "mean=",mean
std = daily_returns['SPY'].std()
print "std=",std

plt.axvline(mean,color='w',linestyle='dashed',linewidth=2)
plt.axvline(std,color='r',linestyle='dashed',linewidth=2)
plt.axvline(-std,color='r',linestyle='dashed',linewidth=2)
plt.show()

# Compute kurtosis
print daily_returns.kurtosis()
```

- So that's it. This line will give you kurtosis of the daily returns.

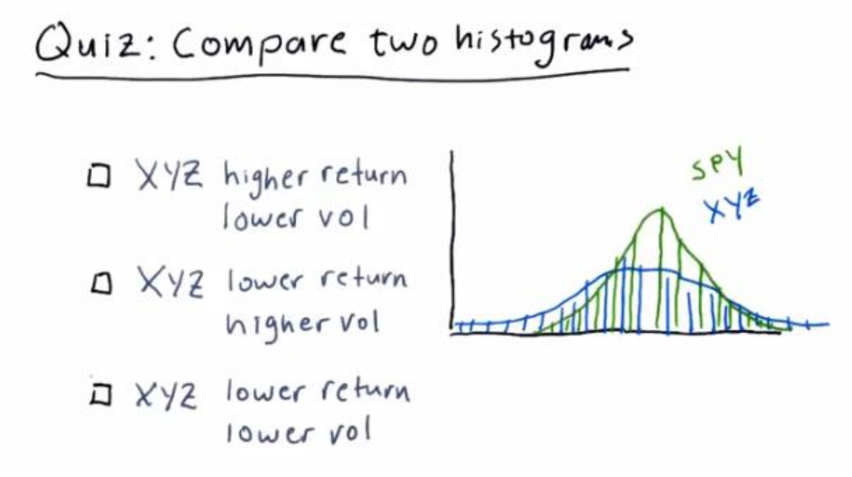


```
mean= 0.000635578332225
std= 0.0133704908994
SPY    3.220278
dtype: float64
```

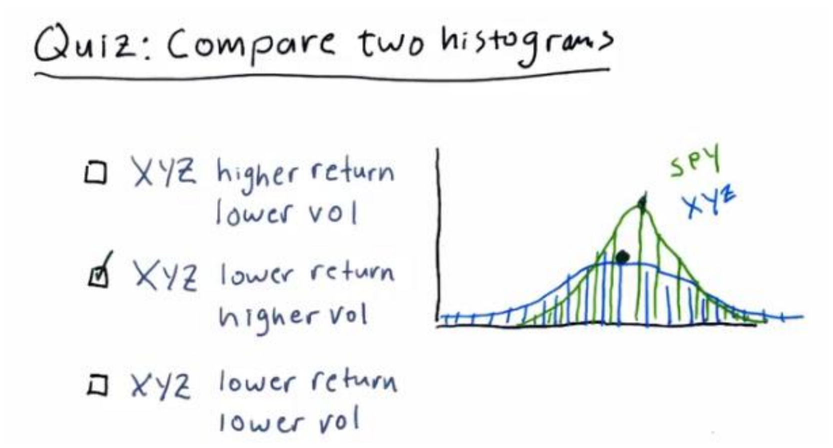


- We get a positive value for the SPY stock, which means we have fat tails. Just for your information, you can also get bincounts using numpy.histogram function. Check instructors notes for more information. Over to you professor.

Compare Two Histograms

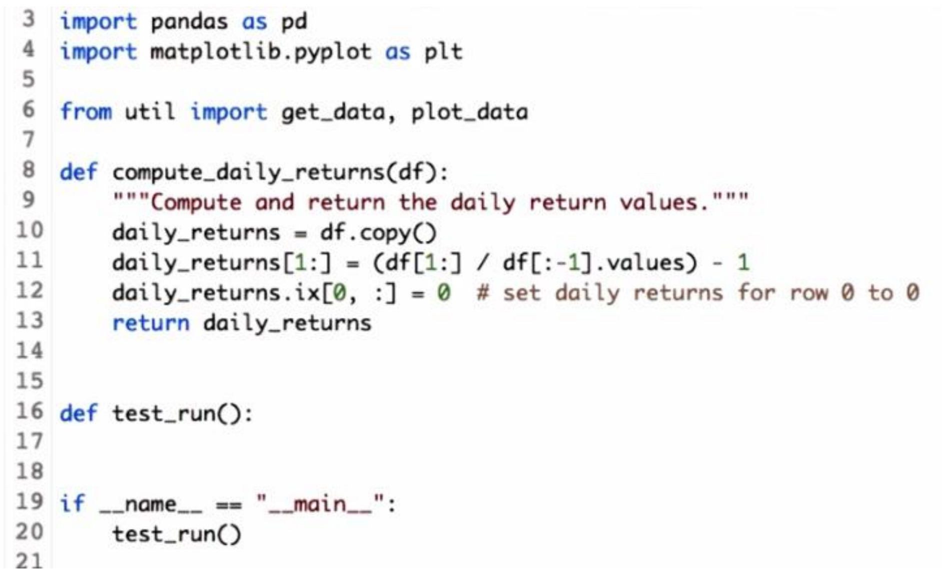


- A common practice in finance is to plot histograms of daily returns of different stocks together and look at them together and assess how they relate to one another.
- So here, we've got a plot of an XYZ stock and SPY or S&P 500. Now take a look and see what difference you can see between them. Now to help out, I'll draw what the underlying shape is looking like.
- Check which answer you think is most correct in terms of volatility and return for XYZ versus SPY.



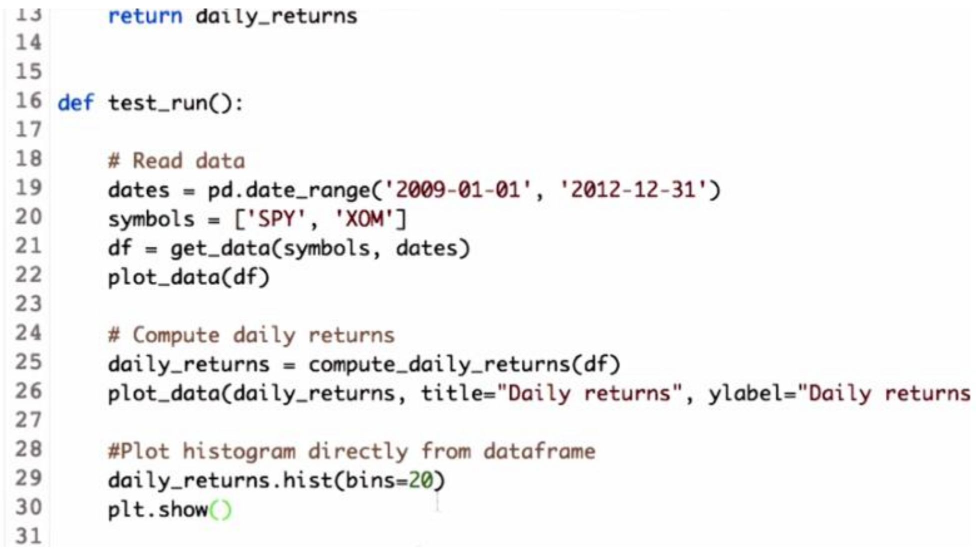
- This is the correct answer. XYZ has a lower return and higher volatility than SPY.
- You can tell that if you look here at the mean of XYZ, you can see it's lower than the mean of SPY. You can see the shoulders are broad on XYZ, meaning it's got a larger standard deviation, and therefore, higher volatility.
- Dave will show you now how to plot histograms like this, right next to one another in Python.

Plot Two Histograms Together



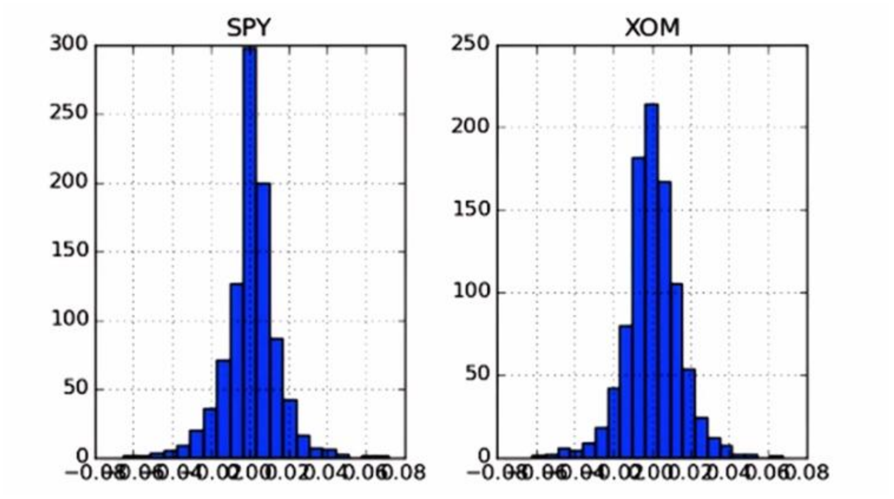
```python
import pandas as pd
import matplotlib.pyplot as plt

from util import get_data, plot_data

def compute_daily_returns(df):
    """Compute and return the daily return values."""
    daily_returns = df.copy()
    daily_returns[1:] = (df[1:] / df[:-1].values) - 1
    daily_returns.ix[0, :] = 0  # set daily returns for row 0 to 0
    return daily_returns


def test_run():


if __name__ == "__main__":
    test_run()
```

- I'm back again. This time with a really small segment. We need to plug two histograms. So first we need the values for two stocks.

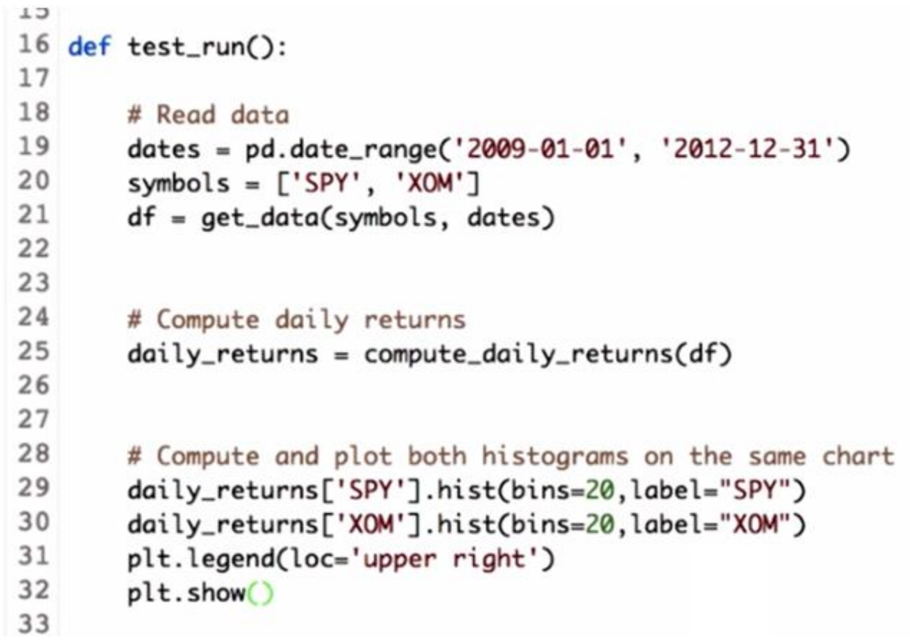


```python
    return daily_returns


def test_run():

    # Read data
    dates = pd.date_range('2009-01-01', '2012-12-31')
    symbols = ['SPY', 'XOM']
    df = get_data(symbols, dates)
    plot_data(df)

    # Compute daily returns
    daily_returns = compute_daily_returns(df)
    plot_data(daily_returns, title="Daily returns", ylabel="Daily returns

    #Plot histogram directly from dataframe
    daily_returns.hist(bins=20)
    plt.show()
```

- We get data for two stocks, which is SPY and XOM. We also go ahead and compute daily returns for each of the stock.
- Note that our daily return data frame will have daily return values for each of the stock prices.
- Now like before, we just call histogram function on the daily return data frame, and let's see what happens. We keep the bins count to be 20. Now let's run this.

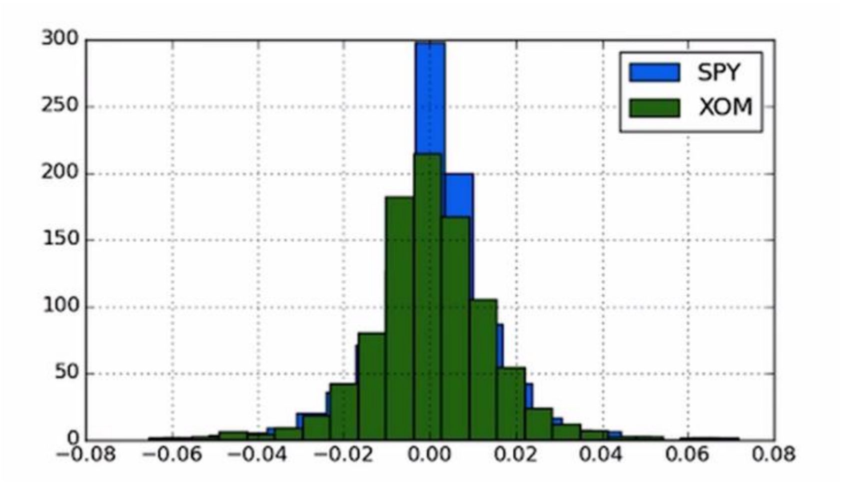


- Okay, so we got two subplots.
- But we go ahead one step and plot them on the same x and y axis so that we can compare the histogram of SPY and XOM.



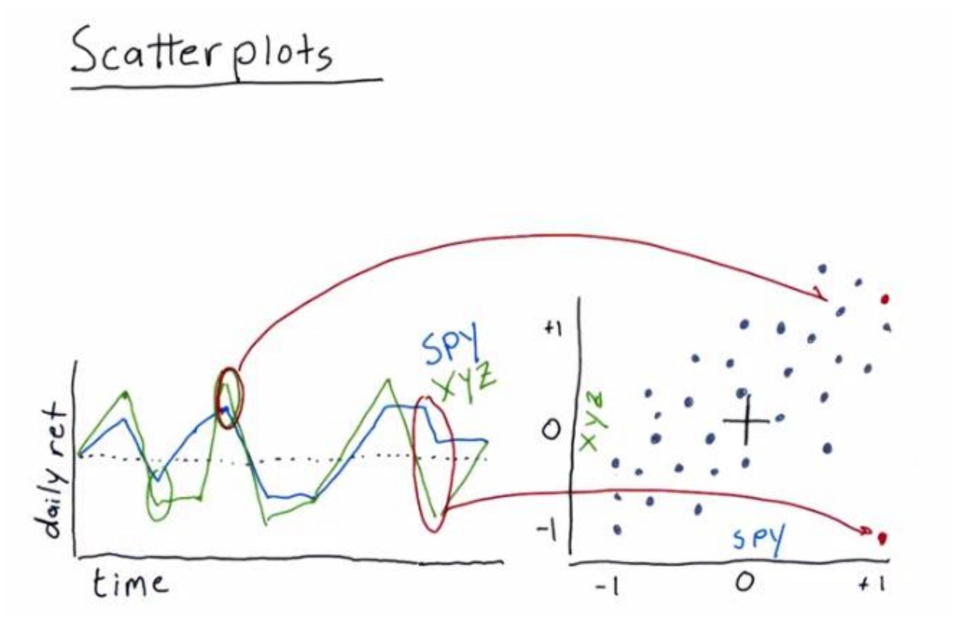
```
def test_run():

    # Read data
    dates = pd.date_range('2009-01-01', '2012-12-31')
    symbols = ['SPY', 'XOM']
    df = get_data(symbols, dates)


    # Compute daily returns
    daily_returns = compute_daily_returns(df)


    # Compute and plot both histograms on the same chart
    daily_returns['SPY'].hist(bins=20,label="SPY")
    daily_returns['XOM'].hist(bins=20,label="XOM")
    plt.legend(loc='upper right')
    plt.show()

```

- To get two histograms on the same x and y axis, we call the histogram functions separately on each of the stocks daily return values.
- We also add the label parameter so that we can differentiate between the histogram of the SPY and XOM. Now, let's run this.



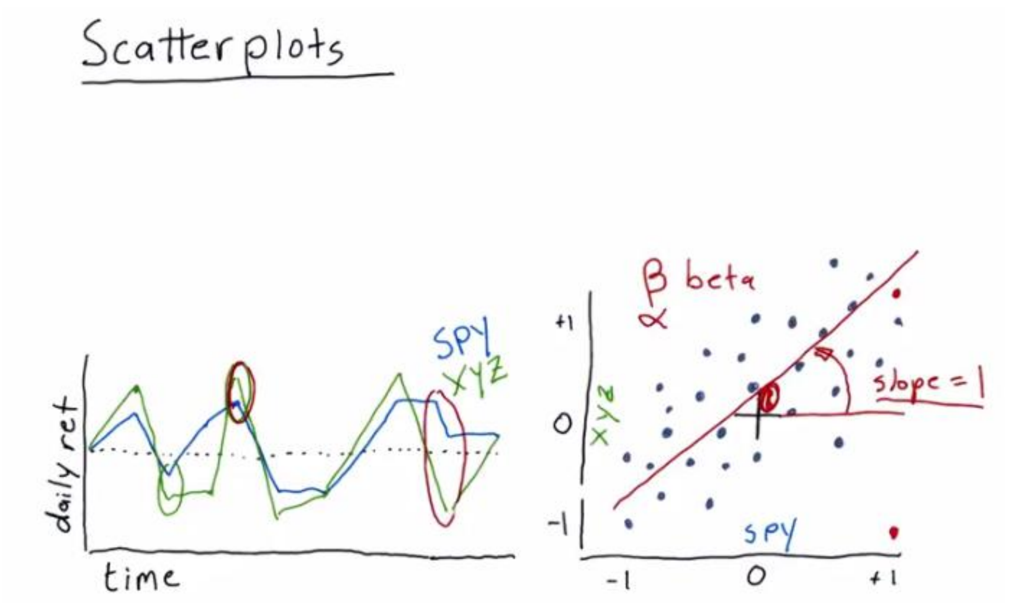
- Here we go. We get two histogram of SPY and XOM on the same X and Y axis. Now you can compare the histogram and see that the teams of XOM is thin as compared to the SPY.
- That's it for this coding segment. I'll be back soon. Over to you professor.

Scatter Plots



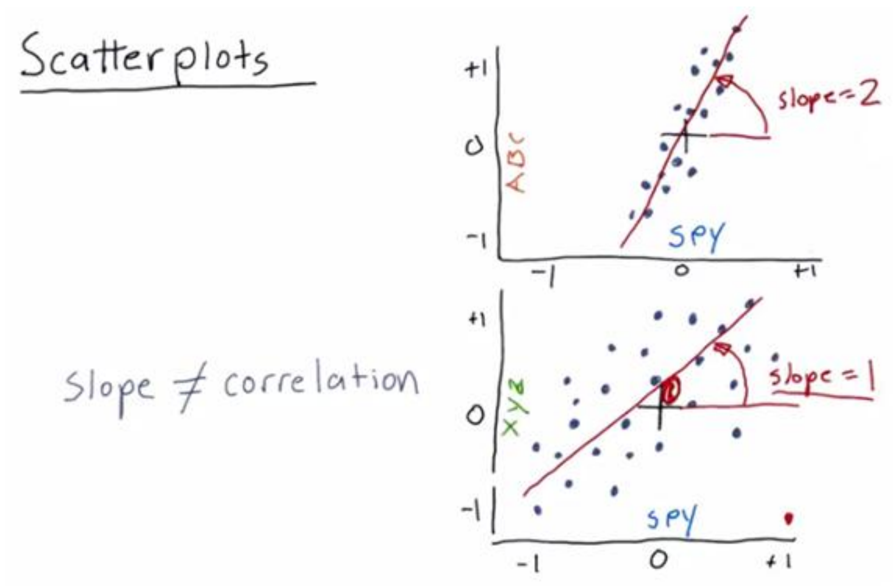
- We're now going to take a look at another way to visualize the differences between daily returns of individual stocks. Let's get back again to our daily return chart.
- We've got S&P 500 here plotted already. And let's compare that to another stock xyz. Now note here that frequently xyz moves in the same direction as spy, but it also sometimes moves a little bit further like in these sections here. We'll be able to visualize those differences in a scatterplot.
- So on a scatterplot, there are a number of individual points or dots. And each one represents something that happened on a particular day. So let's look at this particular day. On this day, spy was positive, close to +1. So we'll look at +1 on spy. And xyz was about +1 as well but a little bit larger. So that day would correspond to a point about there.
- Now we look at each day one by one individually. And populate all of our dots based on what happened each day. Another interesting day is this one where spy and xyz were moving in different directions. So again we had spy in positive territory, but xyz was in negative territory, so that would represent a dot about like that.
- Now if we were to continue this process for very many days over a long period of time, for most stocks a trend appears which is something like this where you can sort of see there's a relationship here, maybe a linear relationship.
- And, uh, however the dots are somewhat scattered. They don't form a perfect line.

Fitting a Line to Data Points



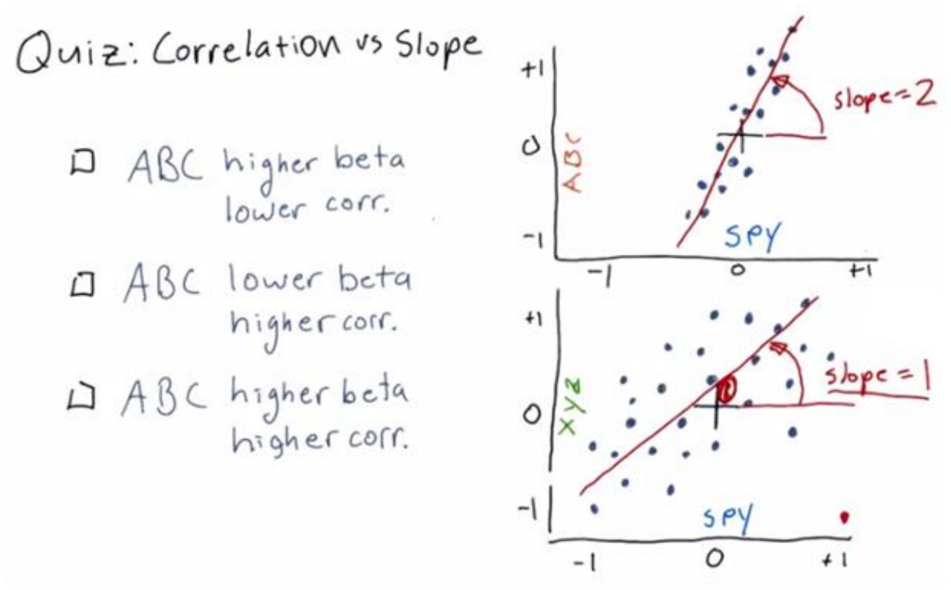
- It is fairly common practice to take this set of data and fit a line to it using linear regression.
- Let's say we got a line, something like that. And to look at the statistics of that linear fit.
- One property is the slope. When we fit a line, what's the slope of that line? Let's assume it turns out to be 1 for this particular stock and its relationship to the S&P 500. This slope, in financial terminology, is usually referred to as beta with this symbol, or just the word beta.
- And what beta means is how reactive is the stock to the market. So if beta is 1, and we have a slope here of 1, it means, on average, when the market goes up 1%, that particular stock also goes up 1%. If we have, say, a higher number, say, 2, that would mean that if the market were to go up 1%, we'd expect on average for that stock to go up 2%.
- There's another factor you can see here when you look at where that line intercepts the vertical axis. That is called alpha. And you've probably heard about alpha in investing circles, and what that means is that this stock is actually on average performing a little bit better than the S&P 500 every day if that number, alpha, is positive.
- If it's negative, it means on average it's returning a little bit less than the market overall. That's how you can plot the data, fit a line to it, and measure a couple aspects of this performance with regard to the market or with regard to some other stock.

Slope Does Not Equal Correlation

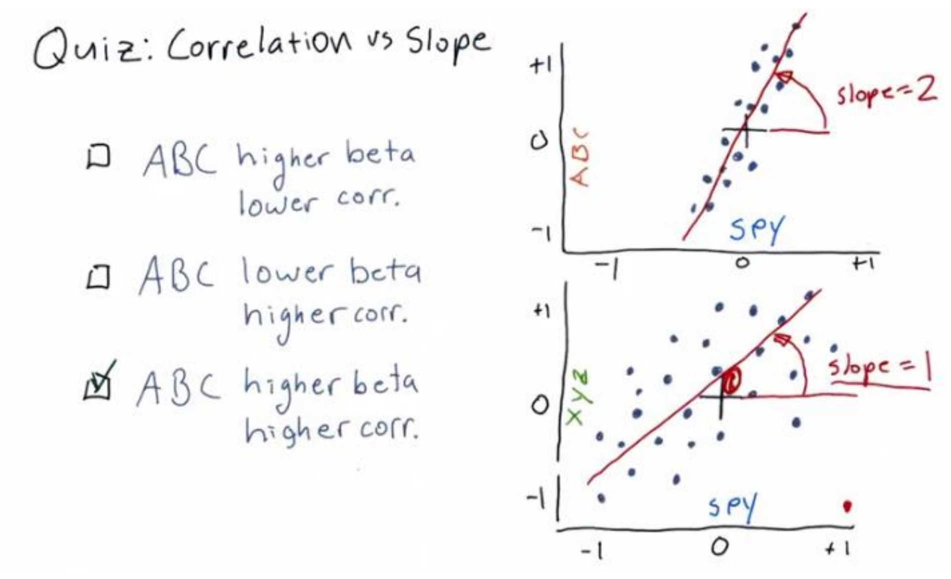


- Now a mistake that a lot of people make is to confound slope with correlation. In other words, if they find that the slope of a line that fits the data is one, that the data are correlated. And that's not true.
- The slope is just the slope. Correlation is a measure of how tightly do these individual points fit that line? So you could have a shallow slope but the data tightly fitting that line, and thus a higher correlation. Or you could have a steeper line and the data fitting that line at a higher correlation.
- Correlation is just a measure of how tightly do those dots fit the line. And you can have a correlation from 0 to 1, zero meaning that the data is not correlated at all. And one meaning that it's very highly correlated.
- Now add a scatter plot for another stock, ABC, and again each one of these dots represents the daily return of SPY versus ABC, and we fit a line to it, and it turns out it's got a slope of two, so now we're going to ask you a couple of questions about it.

Correlation VS Slope

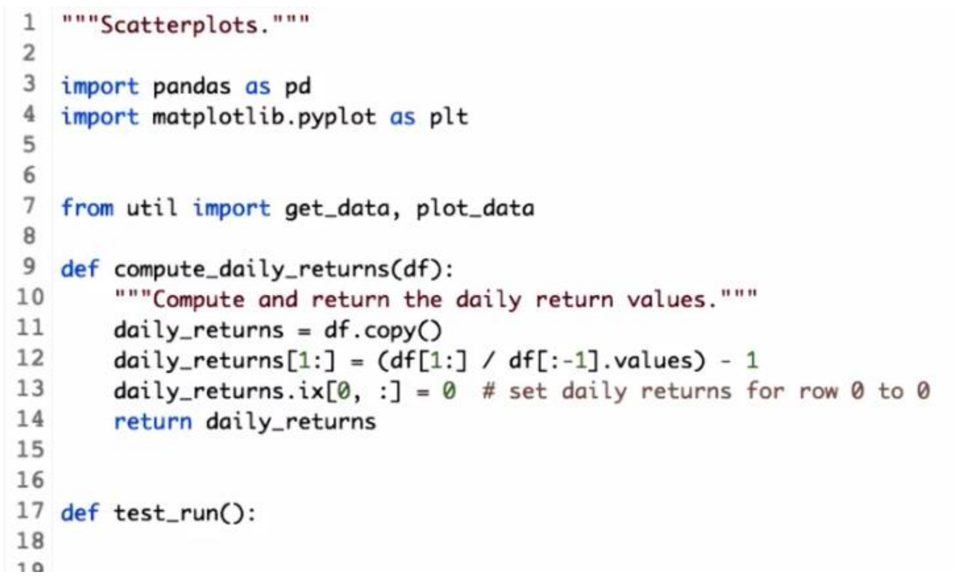

Quiz: Correlation vs Slope
☐ ABC higher beta lower corr.
☐ ABC lower beta higher corr.
☐ ABC higher beta higher corr.
slope=2
ABC
SPY
slope=1
XYZ
SPY
+1
0
-1

- Which of these statements is the most true about the relationship between ABC and XYZ in terms of beta and correlation?


Quiz: Correlation vs Slope
☐ ABC higher beta lower corr.
☐ ABC lower beta higher corr.
☑ ABC higher beta higher corr.
slope=2
ABC
SPY
slope=1
XYZ
SPY
+1
0
-1

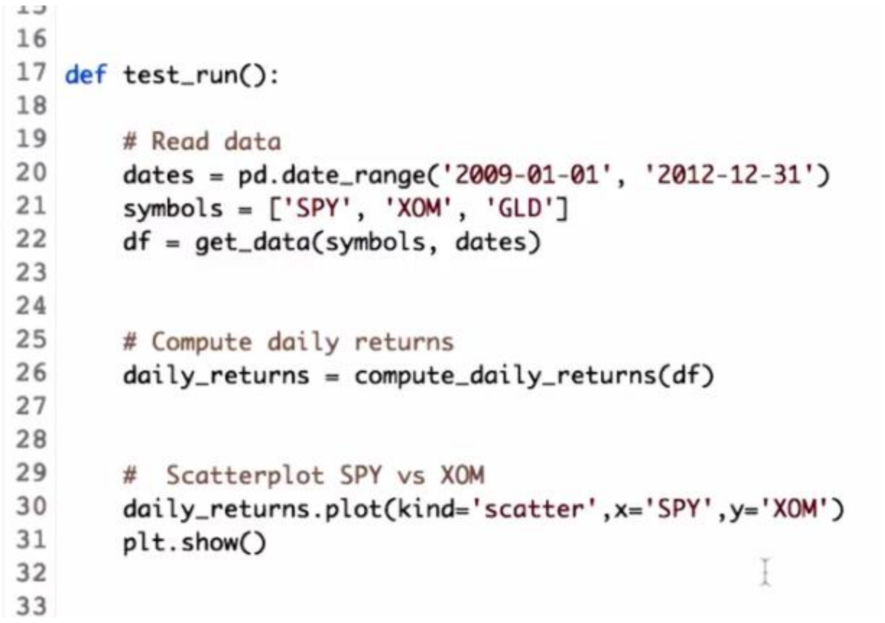
- This is the correct answer. It's got higher beta because the slope is higher and higher correlation because the dots are closer to the line that fits it.
- Now, Dave is going to show you how to create scatter plots and make these measurements in Python. Here's to you, Dave.

Scatter Plots in Python



```
"""Scatterplots."""

import pandas as pd
import matplotlib.pyplot as plt


from util import get_data, plot_data

def compute_daily_returns(df):
    """Compute and return the daily return values."""
    daily_returns = df.copy()
    daily_returns[1:] = (df[1:] / df[:-1].values) - 1
    daily_returns.ix[0, :] = 0  # set daily returns for row 0 to 0
    return daily_returns


def test_run():

```

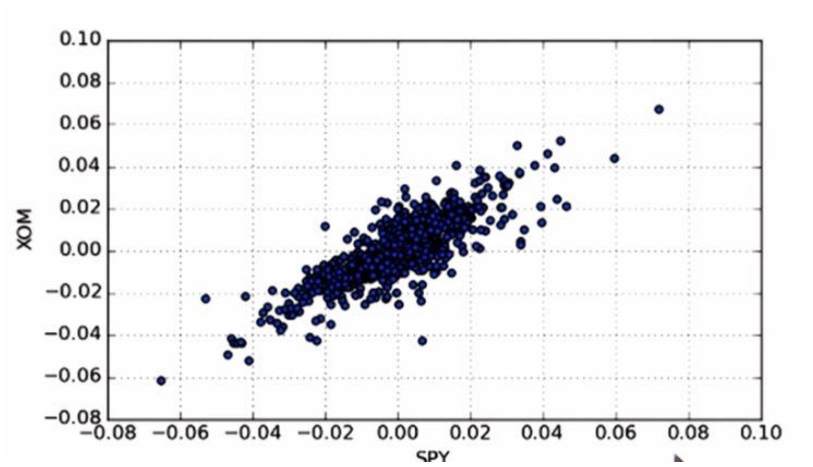
- Thank you, Professor. So I am back with more Python and more graphs. This time, let's scatter some data. I mean, let's learn how to build a scatter plot.
- We will compare scatter plot of SPY versus XOM and SPY versus GLD. So let's read this data and compute daily returns as well.



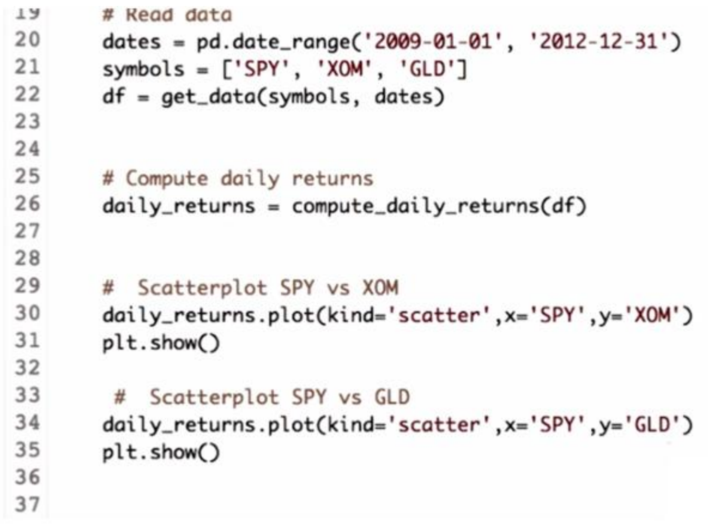
```

def test_run():

    # Read data
    dates = pd.date_range('2009-01-01', '2012-12-31')
    symbols = ['SPY', 'XOM', 'GLD']
    df = get_data(symbols, dates)


    # Compute daily returns
    daily_returns = compute_daily_returns(df)


    #  Scatterplot SPY vs XOM
    daily_returns.plot(kind='scatter',x='SPY',y='XOM')
    plt.show()


```

- As usual, We call get_data function with this symbols and also compute daily returns. Next we first plot scatter plot for SPY versus XOM. Kind parameter of the plot function of the data frame will help us achieve this.

- So we mention we need a scatter plot, but since the data frame daily return has values for three stocks. We have to mention which should be our X axis and which should be our Y axis. As we are plotting SPY versus XOM, we assign X attribute aas SPY and Y attribute as XOM.
- Ready to see the output?

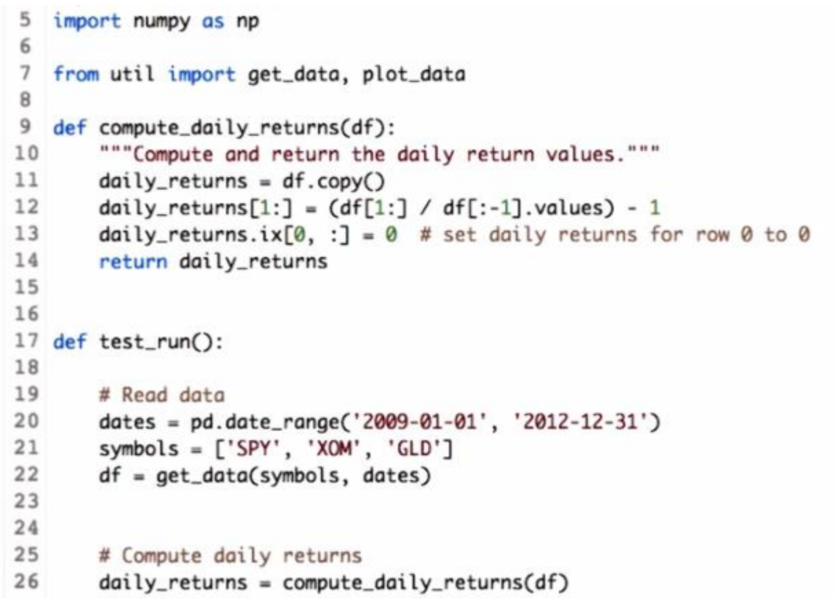


- This is our SPY versus XOM. Now let's similarly plot SPY versus GLD.



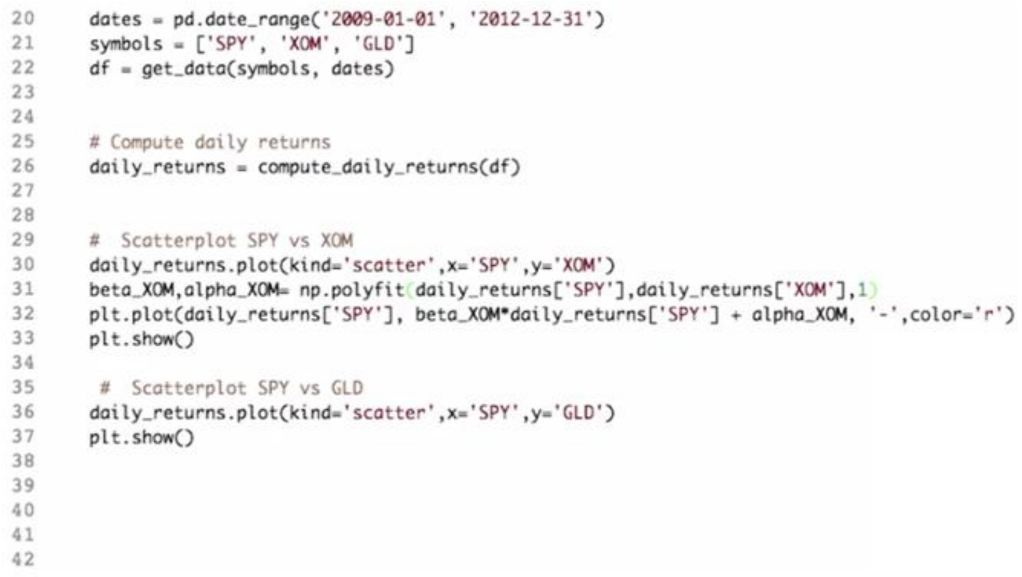
```
# Read data
dates = pd.date_range('2009-01-01', '2012-12-31')
symbols = ['SPY', 'XOM', 'GLD']
df = get_data(symbols, dates)


# Compute daily returns
daily_returns = compute_daily_returns(df)


#  Scatterplot SPY vs XOM
daily_returns.plot(kind='scatter',x='SPY',y='XOM')
plt.show()

 #  Scatterplot SPY vs GLD
daily_returns.plot(kind='scatter',x='SPY',y='GLD')
plt.show()


```

- Since we want GLD on our y-axis, we just replace the y label from XOM to GLD.



```python
import numpy as np

from util import get_data, plot_data

def compute_daily_returns(df):
    """Compute and return the daily return values."""
    daily_returns = df.copy()
    daily_returns[1:] = (df[1:] / df[:-1].values) - 1
    daily_returns.ix[0, :] = 0  # set daily returns for row 0 to 0
    return daily_returns


def test_run():

    # Read data
    dates = pd.date_range('2009-01-01', '2012-12-31')
    symbols = ['SPY', 'XOM', 'GLD']
    df = get_data(symbols, dates)


    # Compute daily returns
    daily_returns = compute_daily_returns(df)
```

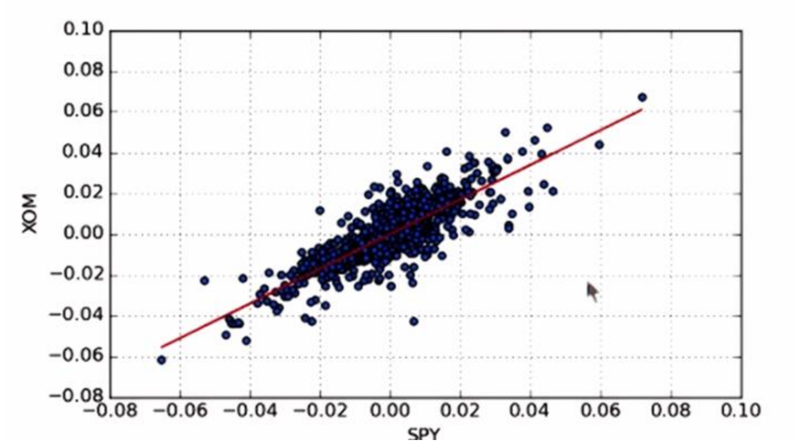
- So here are two scatter plots. SPY versus XOM and SPY versus GLD. But we want to recreate the graph that Professor drew. So we fit a line to the scatter plots.
- For that, we need the help of another good friend of ours. Which is numpy. So let's import it. After importing the numpy library we are all set to fit a line to our scatter plot.
- So we have a set of points, and we want a line which has an equation of degree one. So we go ahead and fit a polynomial of degree one. This is what polyfit function of the numpy does. So let's use it.



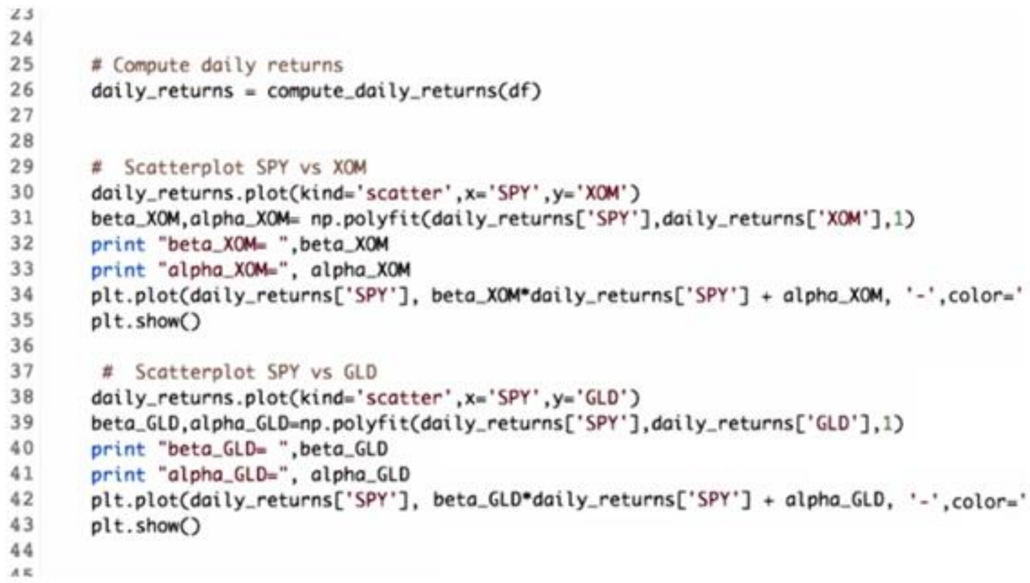
```python
    dates = pd.date_range('2009-01-01', '2012-12-31')
    symbols = ['SPY', 'XOM', 'GLD']
    df = get_data(symbols, dates)


    # Compute daily returns
    daily_returns = compute_daily_returns(df)


    #  Scatterplot SPY vs XOM
    daily_returns.plot(kind='scatter',x='SPY',y='XOM')
    beta_XOM,alpha_XOM= np.polyfit(daily_returns['SPY'],daily_returns['XOM'],1)
    plt.plot(daily_returns['SPY'], beta_XOM*daily_returns['SPY'] + alpha_XOM, '-',color='r')
    plt.show()

     #  Scatterplot SPY vs GLD
    daily_returns.plot(kind='scatter',x='SPY',y='GLD')
    plt.show()
```

- We will first do it for SPY and XOM. The polyfit function needs x-coordinates and y-coordinates to fit a line.
- For us the x-coordinates are the daily return values for SPY and the y-coordinates are daily return values for the XOM.

- The one denotes the degree of our function.
- Calling this function will return two things. The first is the polynomial coefficient and the second is the intercept. Since we have a polynomial of degree 1, it would be of the form y = mx + b. So m is the coefficient and b is the intercept. We name them as beta and alpha.
- Just as Professor explained. Now we finally plot these values. The idea for plotting the line is, for every value of x that is SPY, we find a value of y using the line equation, which is mx + b. This parameter denotes that we want a line plot with the color red.
- Now, let's check our graph.

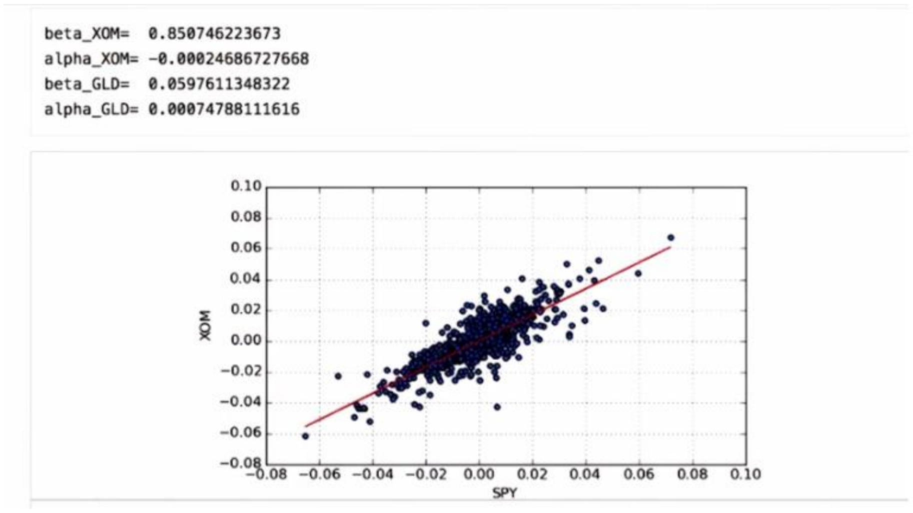


- Here is the fitted line. Let's do it for GLD so that we can compare them both.



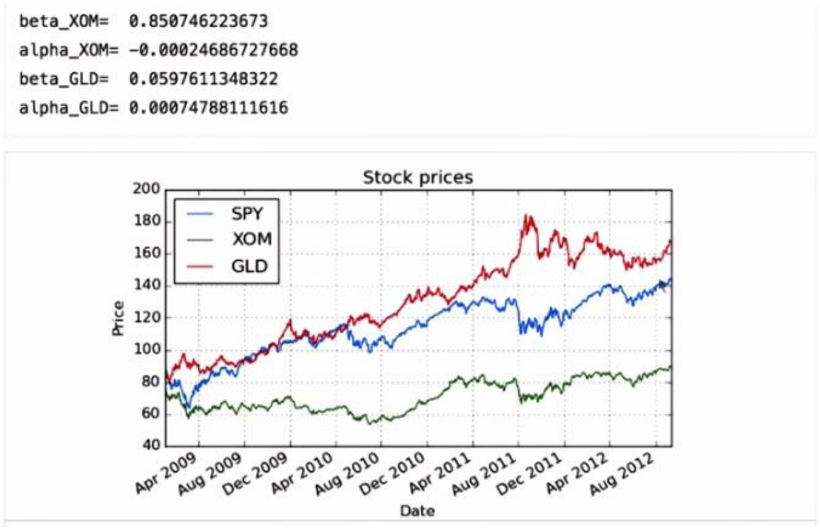
```


    # Compute daily returns
    daily_returns = compute_daily_returns(df)


    #  Scatterplot SPY vs XOM
    daily_returns.plot(kind='scatter',x='SPY',y='XOM')
    beta_XOM,alpha_XOM= np.polyfit(daily_returns['SPY'],daily_returns['XOM'],1)
    print "beta_XOM= ",beta_XOM
    print "alpha_XOM=", alpha_XOM
    plt.plot(daily_returns['SPY'], beta_XOM*daily_returns['SPY'] + alpha_XOM, '-',color='
    plt.show()

     #  Scatterplot SPY vs GLD
    daily_returns.plot(kind='scatter',x='SPY',y='GLD')
    beta_GLD,alpha_GLD=np.polyfit(daily_returns['SPY'],daily_returns['GLD'],1)
    print "beta_GLD= ",beta_GLD
    print "alpha_GLD=", alpha_GLD
    plt.plot(daily_returns['SPY'], beta_GLD*daily_returns['SPY'] + alpha_GLD, '-',color='
    plt.show()

```

- We also print the beta and alpha values for each.



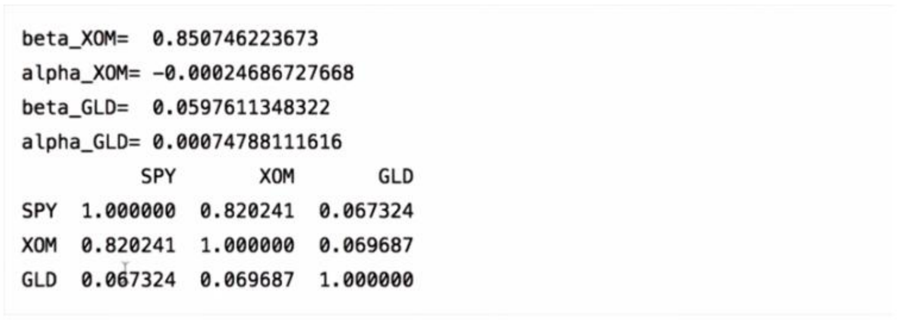
```
beta_XOM=  0.850746223673
alpha_XOM= -0.00024686727668
beta_GLD=  0.0597611348322
alpha_GLD= 0.00074788111616
```

- Now, let's compare the beta values which shows how the stock moved with respect to SPY. You can see that the beta values for the XOM is greater as compared to that of GLD which means that XOM is more reactive to market as compared to GLD.
- On the other hand, the alpha values denote how well it performs with respect to SPY. Numbers over here say that GLD performed better. Let's cross check.



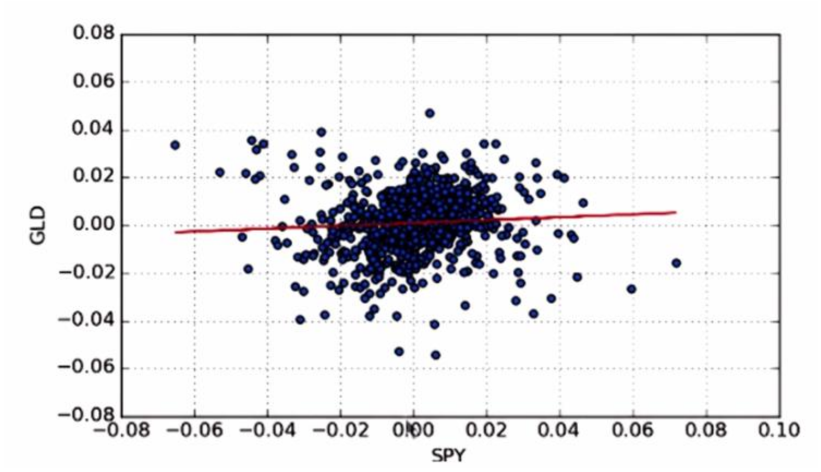
```
beta_XOM=  0.850746223673
alpha_XOM= -0.00024686727668
beta_GLD=  0.0597611348322
alpha_GLD= 0.00074788111616
```

- You can see the upward movement of the GLD as compared to the SPY. One last thing is to find the correlation yet again.



```
beta_XOM=  0.850746223673
alpha_XOM= -0.00024686727668
beta_GLD=  0.0597611348322
alpha_GLD= 0.00074788111616
          SPY       XOM       GLD
SPY  1.000000  0.820241  0.067324
XOM  0.820241  1.000000  0.069687
GLD  0.067324  0.069687  1.000000
```

- The data frame has a function corr which means correlation, and we can define which method to use. We use the method pearson. It is the most commonly used method to calculate the correlation. There are other methods as well, check the instructor's note for more detail.
- We get the output in the matrix format, with correlation of each column with each other column. You can see that the SPY and XOM are highly correlated.
- The value of the correlation for GLD and SPY is very small. Let's check the graph.



- You can observe that the dots do not fit the line closely. And that's why the correlation value for the SPY versus GLD is less as compared to SPY versus XOM.
- For more information on data frame scatter plot and polyfit, check the link in the instructor's notes.

Real World Use of Kurtosis

- As you have seen in this lesson, the distribution of daily returns for stocks and the market look very similar to a Gaussian. This property persists when we look at weekly, monthly, and annual returns as well. If they were really Gaussian we'd say the returns were normally distributed.
- In many cases in financial research we assume the returns are normally distributed. But this can be dangerous because it ignores kurtosis or the probability in the tails.
- In the early 2000s investment banks built bonds based on mortgages. They assumed that the distribution of returns for these mortgages was normally distributed. On that basis they were able to show that these bonds had a very low probability of default.
- But they made two mistakes. First, they assumed that the return of each of these mortgages was independent, and two that this return would be normally distributed. Both of these assumptions proved to be wrong, as massive numbers of homeowners defaulted on their mortgages. It was these defaults that precipitated the great recession of 2008.