Tutorial: Market Simulator

Outline

1. (Review) Install Python and some libraries
2. Download Template File
3. Create a 'market simulator' that builds a portfolio, analyze it, computes expected return.
    1. Create an analyzer:
        - Edit **the analysis.py** file
    2. Create a market simulator on your own
        - Your **Simulator** will use functions from analysis.py which is **[Project 1] a warm-up project.**

Installation:

Step 1: Install your python platform
a): Install Anaconda
Step 2 (later) : Install Market Simulator Templates.

It needs SciPy — so:
Note: The **Anaconda python distribution** includes
* NumPy, Pandas, SciPy, Matplotlib, and Python,
and over 250 more packages available via a
simple "conda install <packagename>"
It also has an IDE.
Instructor got 2.7, and the anaconda distribution of python

To get the appropriate software you'll need:
python (scripting 'programming' language)
sci.py (numerical routines),
num.py (matrices, linear algebra), and
matplotlib (enables generating plots of data)

Installing Python (2.7) via Anaconda:
Anaconda instruction site including lots of libraries with python.
https://docs.continuum.io/anaconda/install

Mac Installation:
1) Instruction that the instructor used:
a) installed anaconda (got required packages)
https://www.continuum.io/downloads (2.7)
includes, sci.py, num.py, and matplotlib
.

Fundamentals

- **Read Data**: Read Stock Data from a CSV File and input it into a pandas DataFrame
  - Pandas.DataFrame
  - Pands.read_csv
- **Select Subsets of Data:** Select desired rows and columns
  - Indexing and slicing data
  - Gotchas: Label-based slicing convention
- **Generate Useful Plots**: Visual data by generating plots
  - Plotting
  - Pandas.DataFrame.Plot
  - Matplot.pyplot.plot

- Scrape S&P 500 ticker list and industry sectors from list of S&P 500 companies on Wikipedia (code provided).
  - https://en.wikipedia.org/wiki/List_of_S%26P_500_companies
- Download **daily close data** for each industry sector from Yahoo finance
  - using pandas DataReader.
- Build a sample Portfolio (in lecture by hand):
- Look at measure**s** of the performance of a portfolio (project 1). We will use the first measure for project 1.
  - **Sharp ratio (in class)**
  - Treynor ratio
  - Jensen's alpha

Goal

- Go from RAW data (adjusted close prices in a .csv file) all the way to visualization

First Something Familiar: Weather Data

- .csv Comma Separated Values of weather conditions from **Oct 2009 to Aug 2017**
- Town of Cary, North Carolina
  - Temperature, pressure, humidity, … lets see
  - Import as “text data”

- Next … stock data.

https://catalog.data.gov/dataset?res_format=CSV&tags=weather

Comma Separated Values (.CSV)

- CSV File
- Header Files
- Lines/Rows of Dates
- Each Element is separated by columns
- Shift-ctrl-down

	A	B	C	D	E	F	G	H	
1	date	temperaturemin	temperaturemax	precipitation	snowfall	snowdepth	avgwindspeed	fastest2minwinddir	fas
2	6/25/12	72	93	0	0	0	6.49	40	
3	7/1/12	75	102.9	0	0	0	4.92	20	
4	7/6/12	71.1	100	0	0	0	3.8	20	
5	7/9/12	73	96.1	0.23	0	0	3.36	180	
6	7/11/12	68	80.1	0.45	0	0	4.03	90	
7	7/21/12	71.1	93	1.09	0	0	7.38	200	
8	7/25/12	70	90	0	0	0	4.03	240	
9	7/27/12	73.9	99	0.14	0	0	6.93	20	
10	7/29/12	66.9	91.9	0	0	0	2.01	100	
11	8/2/12	72	93	0.05	0	0	5.82	180	
12	8/6/12	73.9	93	0.49	0	0	6.26	230	
13	8/10/12	72	87.1	0.13	0	0	8.95	220	
14	8/14/12	66	91	0	0	0	5.59	230	
15	8/24/12	68	77	0.02	0	0	4.7	80	
16	8/26/12	64.9	84	0	0	0	3.13	50	
17	8/28/12	73	87.1	0	0	0	7.61	210	
18	8/30/12	68	89.1	0	0	0	4.47	170	
19	9/2/12	72	88	1.85	0	0	3.58	340	
20	9/14/12	63	80.1	0	0	0	1.57	50	
21	9/25/12	48	77	0	0	0	3.13	210	
22	10/2/12	66.9	88	0.12	0	0	7.61	210	
23	10/7/12	50	64.9	0.23	0	0	6.71	40	
24	10/14/12	44.1	78.1	0	0	0	3.8	230	
25	10/16/12	44.1	70	0	0	0	2.01	280	

What is in a Historical **Stock Data File**?

a) # of employees
b) Date/Time
c) Company Name
d) Price of the Stock
e) Company's Hometown

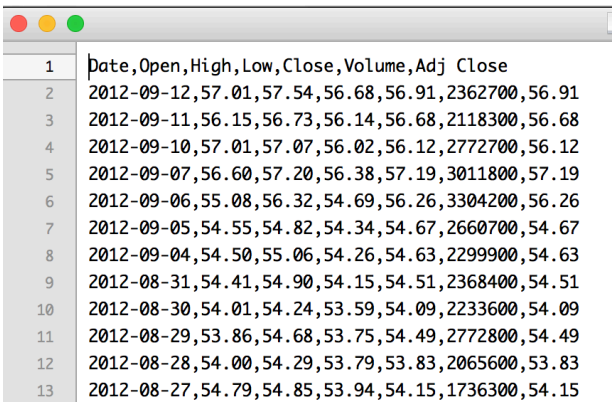
What is in a Historical Stock Data File?

a) # of employees
b) Date/Time
c) Company Name (does not change over time)
d) Price of the Stock
e) Company's Hometown (does not change over time)

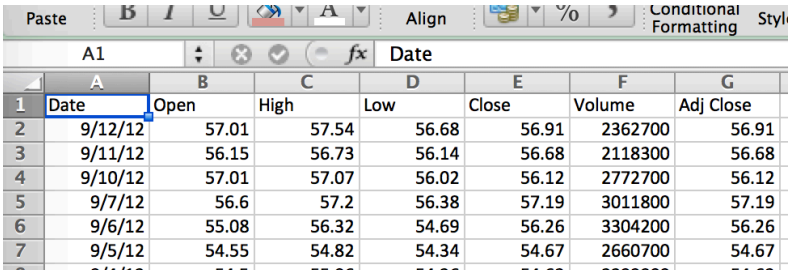
https://finance.yahoo.com/quote/GOOG/history?ltr=1

Comma Separated Values (.CSV)

- Stock Data from Yahoo Finance
- CSV file pulled by panda's (later) DataReader()



```
Date,Open,High,Low,Close,Volume,Adj Close
2012-09-12,57.01,57.54,56.68,56.91,2362700,56.91
2012-09-11,56.15,56.73,56.14,56.68,2118300,56.68
2012-09-10,57.01,57.07,56.02,56.12,2772700,56.12
2012-09-07,56.60,57.20,56.38,57.19,3011800,57.19
2012-09-06,55.08,56.32,54.69,56.26,3304200,56.26
2012-09-05,54.55,54.82,54.34,54.67,2660700,54.67
2012-09-04,54.50,55.06,54.26,54.63,2299900,54.63
2012-08-31,54.41,54.90,54.15,54.51,2368400,54.51
2012-08-30,54.01,54.24,53.59,54.09,2233600,54.09
2012-08-29,53.86,54.68,53.75,54.49,2772800,54.49
2012-08-28,54.00,54.29,53.79,53.83,2065600,53.83
2012-08-27,54.79,54.85,53.94,54.15,1736300,54.15
```



Paste | Align | Conditional Formatting

A1 | Date

	A	B	C	D	E	F	G
1	Date	Open	High	Low	Close	Volume	Adj Close
2	9/12/12	57.01	57.54	56.68	56.91	2362700	56.91
3	9/11/12	56.15	56.73	56.14	56.68	2118300	56.68
4	9/10/12	57.01	57.07	56.02	56.12	2772700	56.12
5	9/7/12	56.6	57.2	56.38	57.19	3011800	57.19
6	9/6/12	55.08	56.32	54.69	56.26	3304200	56.26
7	9/5/12	54.55	54.82	54.34	54.67	2660700	54.67

Stock Data Files

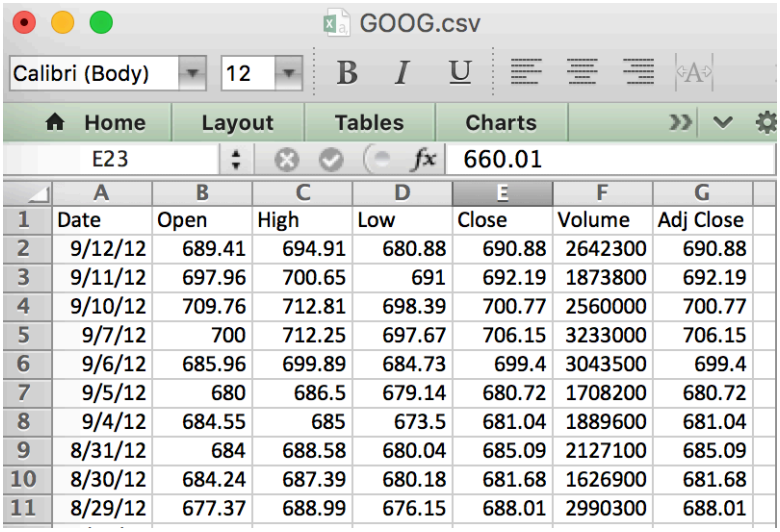
- **Date**
- **Open** – price stock opens at in the morning, it is first price in the day.
- **High** – highest price in the day
- **Low** – lowest price in the day
- **Close** – closing price at 4 PM.
- **Volume** – how many shares traded all together on that day.
- **Adjusted Close** – accounts for splits/and dividends – encapsulates the increase in value if you hold stock for a long time (later).

http://www.investopedia.com/terms/a/adjusted_closing_price.asp

https://finance.yahoo.com/quote/GOOG/history?ltr=1

GOOG.csv (from Yahoo).

- Newer dates on top, older descending.



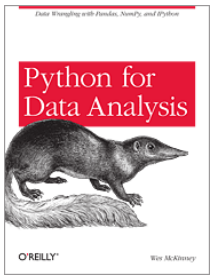
	A	B	C	D	E	F	G
1	Date	Open	High	Low	Close	Volume	Adj Close
2	9/12/12	689.41	694.91	680.88	690.88	2642300	690.88
3	9/11/12	697.96	700.65	691	692.19	1873800	692.19
4	9/10/12	709.76	712.81	698.39	700.77	2560000	700.77
5	9/7/12	700	712.25	697.67	706.15	3233000	706.15
6	9/6/12	685.96	699.89	684.73	699.4	3043500	699.4
7	9/5/12	680	686.5	679.14	680.72	1708200	680.72
8	9/4/12	684.55	685	673.5	681.04	1889600	681.04
9	8/31/12	684	688.58	680.04	685.09	2127100	685.09
10	8/30/12	684.24	687.39	680.18	681.68	1626900	681.68
11	8/29/12	677.37	688.99	676.15	688.01	2990300	688.01

https://finance.yahoo.com/quote/IBM/history

- Adjusted Close – adjusts / accounts for stocks splits and dividend payments.
- On the Current Day – Adjusted Close and Close are always the same.
- Previous Days:
  - But as we go back in time start they to differ they are not always the same.
  - Actual Return is not captured by the closing price, need to use adjusted close on historical data.

Pandas: Included in Anaconda

- https://en.wikipedia.org/wiki/Pandas_(software)
- Developed by Wes McKinney while at AQR Capital Management to analyze financial data
  - Open Source.
  - Numerical Tables and **Time Series**
  - **A Key Element : Data Frames**
    - **Slicing**
  - Panel Data



Store Portfolio in a Panda Data Frame

- Want: <Symbols> vs Time
- Includes a set of equities (ownership)
  - Exchange Traded Fund (ETF)
  - SPY 500
    - Tracks the index S&P 500 Index.
  - Russell 1000
  - AAPL – apple
  - GOOG – Google
  - Other: securities (government)
- **NaN**
- **https://en.wikipedia.org/wiki/Google**
  - **Initial public offering (IPO) - August 19, 2004.**

symbols

time

	SPY	AAPL	GOOG	GLD
2010-01-04				
2010-01-05				
2010-01-06				
2010-01-07				
2010-01-08				
2010-01-11				
2010-01-12				
2010-01-13				
2010-01-14				
2010-01-15				
2010-01-19				
2010-01-20				
2010-01-21				
2010-01-22				
2010-01-25				

https://finance.yahoo.com/quote/GOOG/history?ltr=1

Warm-up: Reading into a Data frame

- Interactively
  - Import pandas
  - Rename it to pd
- Read it in.
- First column is index helping you to access rows.
- SPY, AAPL, GOOG, GLD

```
{ingrid:632} python
Python 2.7.11 |Anaconda 4.1.0 (x86_64)| (default, Jun 15 2016, 16:09:16)
[GCC 4.2.1 (Based on Apple Inc. build 5658) (LLVM build 2336.11.00)] on darwin
Type "help", "copyright", "credits" or "license" for more information.
Anaconda is brought to you by Continuum Analytics.
Please check out: http://continuum.io/thanks and https://anaconda.org
>>> import pandas as pd
>>> df = pd.read_csv("data/AAPL.csv")
>>> print df.head()
         Date    Open    High     Low   Close    Volume  Adj Close
0  2012-09-12  666.85  669.90  656.00  669.79  25410600     669.79
1  2012-09-11  665.11  670.10  656.50  660.59  17987400     660.59
2  2012-09-10  680.45  683.29  662.10  662.74  17428500     662.74
3  2012-09-07  678.05  682.48  675.77  680.44  11773800     680.44
4  2012-09-06  673.17  678.29  670.80  676.27  13971300     676.27
>>> print df
           Date    Open    High     Low   Close    Volume  Adj Close
0    2012-09-12  666.85  669.90  656.00  669.79  25410600     669.79
1    2012-09-11  665.11  670.10  656.50  660.59  17987400     660.59
2    2012-09-10  680.45  683.29  662.10  662.74  17428500     662.74
3    2012-09-07  678.05  682.48  675.77  680.44  11773800     680.44
4    2012-09-06  673.17  678.29  670.80  676.27  13971300     676.27
5    2012-09-05  675.57  676.35  669.60  670.23  12013400     670.23
6    2012-09-04  665.76  675.14  664.50  674.97  13139000     674.97
7    2012-08-31  667.25  668.60  657.25  665.24  12082900     665.24
8    2012-08-30  670.64  671.55  662.85  663.87  10810700     663.87
9    2012-08-29  675.25  677.67  672.60  673.47   7243100     673.47
10   2012-08-28  674.98  676.10  670.67  674.80   9550600     674.80
11   2012-08-27  679.99  680.87  673.54  675.68  15250300     675.68
12   2012-08-24  659.51  669.48  655.55  663.22  15619300     663.22
13   2012-08-23  666.11  669.90  661.15  662.63  15004600     662.63
14   2012-08-22  654.42  669.00  648.11  668.87  20190100     668.87
15   2012-08-21  670.82  674.88  650.33  656.06  29025700     656.06
16   2012-08-20  650.01  665.15  649.90  665.15  21906600     665.15
17   2012-08-17  640.00  648.19  638.81  648.11  15812900     648.11
```

Exercises

Exercise 1.

- Read in the entire CSV file in a function
  - Print it out.

Exercise 2.

- Read in the entire file in a function
  - Print out a selection of file
    - Top 5 lines : .head()
    - Bottom 5 lines: .tail()

def -- Make it a function

```
1  import pandas as pd
2
3
4  def test_run():
5      df = pd.read_csv("data/AAPL.csv")
6      print df #print entire dataframe
7
8
9  if __name__ == "__main__":
10     test_run()
```

- simple-frame.py
  - Entire frame
  - **Try: printing -** df.head(), df.tail()
- **Question**: Print last 5 lines?

- Only print top 5 line of data frame
  - print df.head()
- Only print bottom 5 lines of data frame
  - print df.tail()

Print out a subset of columns, and/or rows:

- **Slicing**: Only print rows between index 10, 20 (not inclusive)
  - print df[10:21]
  - print df[:21]
  - print df[['Date','High']].values[5]

Computation on CVS File

- From the file, find out maximum closing price.
  1. Read the file into a data frame
     - **Now** - SPY.csv
     - **Later** – any symbol.
  2. Process the Column 'Close'
  3. Use pandas function .max() to return max.

Compute Max Closing Price get_max_close(**symbol**)

```
import pandas as pd

def get_max_close(symbol):
        """Return the maximum closing value for stick indicated by symbol.

        Note: Data for stock is stored in file: data/<symbol>.csv
        """
        df = pd.read_csv("data/{}.csv".format(symbol))  # read in data
        return df['Close'].max()                        # compute & return max

def test_run():
        """Function called by Test Run."""
        for symbol in ['AAPL', 'IBM']:
                print "Max close"
                print symbol, get_max_close(symbol)


if __name__ == "__main__":      # if run standalone
        test_run()
```

1a-maxclosingprice.py

https://pyformat.info/

Exercises

- Calculate the mean volume.
- Calculate the max adjusted close.
- **Challenge**: Return date(s) when :
  - closing price is different from the adjusted price?
  - IBM

1b-meanvolume-quiz.py

Plotting maplotlib

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

def test_run():
        df = pd.read_csv("data/AAPL.csv")  # read in data
        print df ['Adj Close']
        df ['Adj Close'].plot()
        plt.show()      # must be called to show plots

if __name__ == "__main__":      # if run standalone
        test_run()
```

2a-1column-plots.py

http://matplotlib.org/users/pyplot_tutorial.html#working-with-text

Plot **2 Columns** in a single Plot

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

def test_run():
        df = pd.read_csv("data/AAPL.csv")  # read in data
        #print df ['Adj Close']
        df [['Close','Adj Close']].plot() # double square brackets.
        plt.show()      # must be called to show plots

if __name__ == "__main__":      # if run standalone
        test_run()
```

2b-2column-plots.py

Coming UP.

- Restrict Data Ranges (e.g., specific date range)? (join)
- Drop Missing Data Rows
- Join Data Incrementally, column by column

Want to get a frame with Closing date of Different Stocks.

```
{ingrid:503} python 3b-multiple.py
            Adj Close    GOOG     IBM     GLD
2010-01-22     104.34  550.01  119.61  107.17
2010-01-25     104.87  540.00  120.20  107.48
2010-01-26     104.43  542.42  119.85  107.56
```

Only on trading days …

How many days were US Stocks Traded in 2014 (over an entire year)

a) 365
b) 260
c) 252

How many days were US Stocks
Traded in 2014 (over an entire year)

a) 365

b) 260 (52x5) But there are also holidays …

c) 252

Steps: Building a DataFrame

1. DF1 = First build a data frame by **specifying the date range**.
   – Includes weekend dates (markets are not open).
2. DF2 = SPY = Load in SPY data (adjusted close) into a separate data frame (all data and prices).
   – Only trading days (market open) in DF2.
3. **Join DF2 and DF 1** – join so that **only dates that are present** in 'both' frames (it eliminates the weekends in Data frame 1).
4. Additional Joins with other 'symbol' that we want to add, IBM, GOOG.

Steps 0-2: Specifying the Data Range

- **Step 0:**
- **Step 1: Create a list of data time index objects**
  - `dates = pd.date_range(start_date, end_date)`
  - Check it out (print).
    - List of data time index objects
      - Dates[0] (dates with time stamp)
      - Dates[1]
- **Step 2. Index it by dates** instead of integer by specifying index and setting it to 'dates'
  - index = dates.
  - NOTE seen the default of integers already …

3a-simple-join.py

Step 3: Combine the data frames with Joining Frames

a) df2: Create SPY date frame w/ SPY data

b) Combine date frames via join.

- df1: Empty date frame with a **date range**
- df2_SPY Populated date frame (only trading days)
- Join: **left** join
  - df1.join(df2_SPY)
  - Only SPY row are retained.
- ? No values from SPY??

- dfSPY is indexed by integers by default, change index to dates by index_col
  - index_col="Date"

- Multiple Stocks from a list
  - symbols = ['GOOG', 'IBM', 'GLD']
  - For loop iterating through symbols

```
pd_read_csv("data/{}.csv".format(symbol),
    index_col='Date',
    parse_dates=True,
    Usecols=['Date',Adj Close'],
    na_values=['nan'])
```

  - … overlap of Adj Close column
    - Rename the column to stock symbol instead.

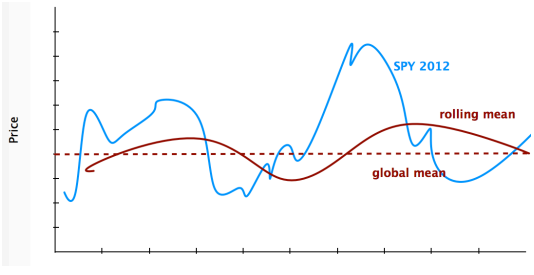
Exercise:

- Utility Functions to **read in data no NaNs.**

```
3  import os
4  import pandas as pd
5
6  def symbol_to_path(symbol, base_dir="data"):
7      """Return CSV file path given ticker symbol."""
8      return os.path.join(base_dir, "{}.csv".format(str(symbol)))
9
10
11 def get_data(symbols, dates):
12     """Read stock data (adjusted close) for given symbols from CSV files."""
13     df = pd.DataFrame(index=dates)
14     if 'SPY' not in symbols:  # add SPY for reference, if absent
15         symbols.insert(0, 'SPY')
16
17     for symbol in symbols:
18         # TODO: Read and join data for each symbol
19
20     return df
21
```

Re-Cap: Last Week

- Worked on board … on code.
- Compute / Code financial statistics in pandas and numPY:
  - Global Statistics
    - Mean
    - Median
    - Standard Deviations
  - Rolling Statistics
    - Rolling mean
      - Representation of underlying value of a stock
    - Rolling standard deviation
      - deviate from the mean (buy and sell signal)



- Bollinger Bands
  - Upper band :
    - rolling mean + 2 * rolling StdDev
  - Lower band :
    - rolling mean – 2 * rolling StdDev

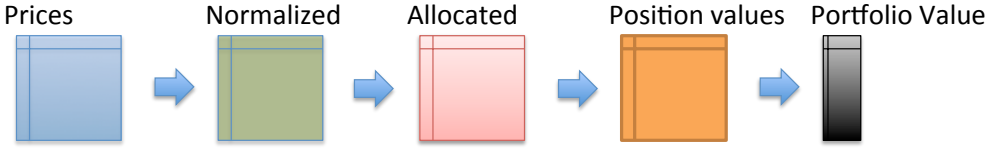
https://en.wikipedia.org/wiki/Bollinger_Bands

Get the Daily Total Value of the Portfolio

- **Step 1**: Prices Data Frame index by dates
- **Step 2**: Normalize by First Row
  - Normed = prices/priced[0]
- **Step 3**: Multiply by allocation (a vector)
  - Allocated = Normed * allocs
- **Step 4**: Position values = worth each day
  - Pos_vals = Allocated * start_val
- **Step 5**: Daily Total Value of Portfolio
  - Port_val = Pos_vals.sum(axis = 1)

```
Given:

start_val   = $1,000,000
start_date  = 2011-01-01
end_date    = 2011-12-31
symbols =['SPY','XOM',
          'GOOG', 'GLD']
allocs = [0.4,0.4,0.1,0.1]
```



Daily Return on the portfolio value

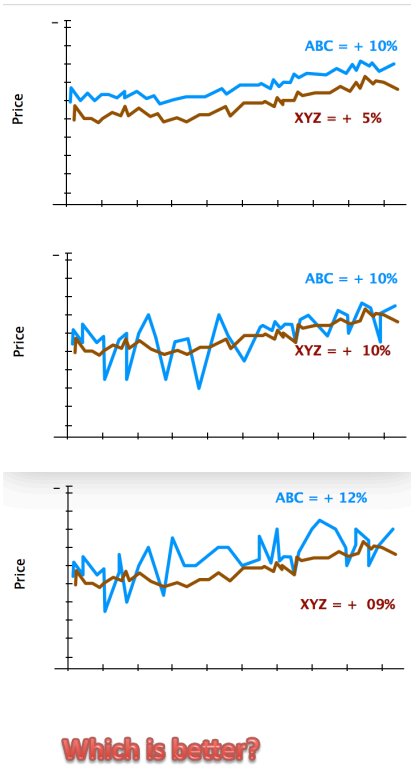
- Daily return[t] = (prices[t]/prices[t-1]) -1
  - Now on port_val (instead of prices).
  - Observation: 1st value is always 0
    - daily_rets = daily_rets[1:]

Statistics on the Portfolio

- Cumulative Returns
  - Form beginning to end (last value/initialial val) -1
    - cum_ret = (port_val[-1]/port_val[0]) - 1
- Average Daily Returns
  - daily_rets.mean()
- Standard Deviation of Daily Return
  - Daily_rets.std()
- Sharpe Ratio

Sharpe Ratio

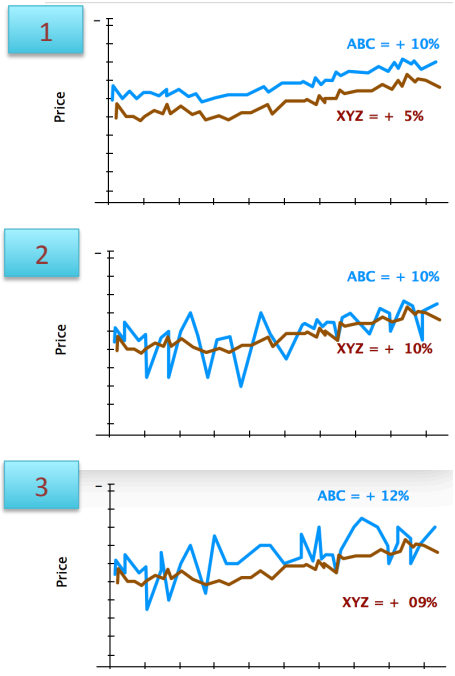
- Considers our return in the context of risk
- Risk is volatile (standard deviation)
- Adjust our return in return for the risk
- Volatility
  - Measured by standard deviation



Which is better?

Sharpe Ratio

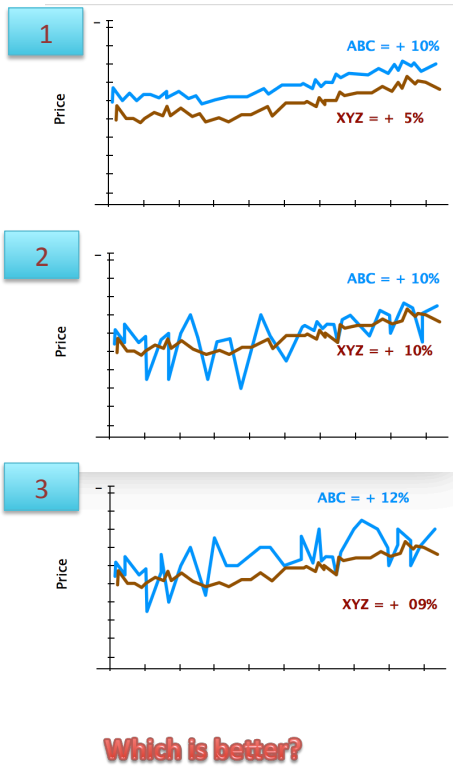
- Considers our return in the context of risk
- Risk is volatile (standard deviation)
- Adjust our return in return for the risk
- Volatility
  - Measured by standard deviation



Which is better?

Sharpe Ratio

1. Higher Returns is Better
2. Less Volatility/Less Risk is Better
3. Not Enough Information
   - Returns: ABC > XYZ
   - Volatility ABC > XYZ
   - **ABC is higher returns, but more risk**



Which is better?

Sharpe Ratio

- Adjusts return for risk
- A quantities way to assess portfolio
  - 1. ABC is better because same volatilty but higher returns
  - 2. same returns but XYZ has lower risk so XYZ is better
  - 3. Sharpe Ratio may give as a number to determine which is better
- Sharpe ratio also considers (comparative)
  - Risk free rate of returns
    - Bank account or treasure note
  - Lately risk free return is 0, bank interest rate is 0, or or close to 0

Which Formula is Best?

- R_p : Portfolio Return
- R_f : Risk Free Rate of Return
- σ_p : Standard Deviation of Portfolio Return

a) $R_p - R_f + \sigma_p$
b) $R_f / R_f - \sigma_p$
c) $(R_p - R_{f)} / \sigma_p$

General Form of the Sharpe Ratio

Computing Sharpe Ratio

- SR (expected value)
  = E [Rp – Rf]/std[Rp-Rf]
- Expected value → mean over time:
  = mean(daily_rets – daily_rf)/std(daily_rets – daily_rf)

- What is the risk free rate?
  – LIBOR (London Inter Bank Offer Rate)
  – Interest Rate 3 months Treasury Bill
  – 0%! Short Cut.

Computing Sharpe Ratio

- SR (expected value)
  = E [Rp – Rf]/std[Rp-Rf]
- Expected value → mean over time:
  = mean(daily_rets – daily_rf)/std(daily_rets – daily_rf)

- Risk Free Rate not given on a daily bases
  - LIBOR
  - Annual/6 month bases
  - Short Cut –
    - Convert Annual to a daily amount
    - Example:
      - Annual Rate: 0.1 per year Risk Free RATe
      - Total Value at end of year: 1.0 * 0.1
      - What is the Interest Rate per Day:
        - » Daily_RF = SQRT_252(1.0 + 0.1) – 1 → 0.0 (approximation)
  - Constant Standard Deviation of a Constant

Sample Frequency

- SR can vary depending on how frequently we sample the data (need an adjustment factor to convert between different sampling)
  - Annual (initial vision of SR)
  - Monthly
  - Daily

$SR_{annualized} = k * SR$

k = sqrt (# samples per year)

Sample Frequency

- SR can vary depending on how frequently we sample the data (need an adjustment factor to convert between different sampling)
  - Annual (initial vision of SR)
  - Monthly
  - Daily

Daily k = sqrt (252)
Weekly k = sqrt (52)
Montly K = sqrt (12)

$SR_{annualized} = k * SR$

k = sqrt (# samples per year)

ReCap: Sharpe Ratio for Daily Returns

- SR

  = sqrt(252)

  * (mean(daily_rets – daily_rf)

  / std(daily_rets-daily_rf))

Quiz: What is the Sharpe Ratio

- Given:
  - 60 days of data
  - Average daily return = 0.001 (10 bases points)
  - Dailyrisk free return = 0.0002 (2 bases points)
  - Std daily return = = 0.001
- What is the Sharpe Ratio?

Quiz: What is the Sharpe Ratio

- Given:
  - 60 days of data
  - Average daily return = 0.001 (10 bases points)
  - Dailyrisk free return = 0.0002 (2 bases points)
  - Std daily return = = 0.001
- What is the Sharpe Ratio?
- = Sqrt(252) * mean(Rp-Rf)/Std(Rp)
  - = Sqrt(252) * (10-2)/10 = 12.7

Python

- sharpe_ratio =

  np.sqrt(samples_per_year)

  * np.mean(daily_rets - daily_rf) / std_daily_ret