Day 1 WS 2 -- Stock Market and Sage

Introduction to Sage

1. Sage is free open source mathematical software (show web page).
2. There are about 100 developers (show developer map).
3. You can use Sage via a web browser or install it on your own computer.
4. Sage itself is:
    1. A distribution of about **5 million** lines of open source free software,
    2. A large new library, and
    3. Interfaces to almost all existing math software.
5. Documentation:
    1. The tutorial
    2. The reference manual
    3. Introspection (help from within Sage)
    4. The Python language -- you'll be able to do much more if you know some programming and learn some Python. This isn't a prerequisite.
6. How to use the notebook:
    1. Press shift-enter
    2. Publishing worksheets and rating them.

Problem 1: Printing Money

The point of this exercise is to give you some practice doing basic

arithmetic with Sage, and also to show you how money and value are counterintuitive. 

1. I buy 100 shares of SIMUW for $100.
2. You think SIMUW is going up and buy 10 shares for $115.
3. Someone else things you're very clever and also things the value of SIMUW will go up further, so purchases 10 more shares for $140, feeling they have made a good buy.
4. Since people are buying shares of SIMUW for $140/share, I feel my shares are worth $140/each.

Questions:

1. How much actual money has each participant -- me, you, and somebody else -- put into the market?
2. Assuming shares are now seeling for $140/each, how much total money do we feel that our shares are worth?
3. Where did that extra money come from?

```
100*140   # press shift enter
```

The [1929 stock market] crash followed a speculative boom that had taken hold in the late 1920s, which had led hundreds of thousands of Americans to invest heavily in the stock market, a significant number even borrowing money to buy more stock. By August 1929, brokers were routinely lending small investors more than 2/3 of the face value of the stocks they were buying. Over $8.5 billion was out on loan, more than the entire amount of currency circulating in the U.S.. The rising share prices encouraged more people to invest; people hoped the share prices would rise further. Speculation thus fueled further rises and created an economic bubble. [...] Most economists view this event as the most dramatic in modern economic history. -- from Wikipedia

1. I buy 100 shares of SIMUW for $100.
2. You think SIMUW is going down and so do I, so you manage to buy 10 shares for $85.
3. You're not feeling so happy about your purchase, so you put them on sale for $70. Someone else thinks that's a good buy, so

buys your 10 shares for $70.
4. Since people are buying shares of SIMUW for $70/share, I feel my shares are worth $70/each now.

Questions:

1. How much actual money has each participant -- me, you, and somebody else -- put into the market?
2. Assuming shares are now seeling for $70/each, how much total money do we feel that our shares are worth?
3. Where did that extra money go?

Problem 2: Plotting Functions

Use Sage to draw plots of the following functions: $x^2 + 2$, $\sin(x)$, $\cos(x^2 \sin(x))$ on various intervals

```
var('x')
plot(sin(x), (x, -10, 10))
```

Problem 3: Draw 3d Plots

Use Sage to draw plots of the following functions: $x^2 + y^2$, $\sin(xy)$, and $x^2 - \sin(x/y)$.

```
var('x,y')
plot3d(x^2 + y^2, (x, -10, 10), (y,-10,10))
```

Problem 4: Compute some basic statistics

Use Sage TimeSeries to compute the mean and variance of a list of numbers:

The **mean** of numbers $x_1, \ldots, x_n$ is the sum of the x_i divided by n. It's just the "average".

The **variance** of numbers $x_1, \ldots, x_n$ is the sum of the squares of the differences of the x_i from the mean (divided by $n-1$). It measures by how much the numbers **vary**.

The code

```
v = TimeSeries([ random() for i in [1..100] ])
```

will make a series of 100 distinct randomly chosen values. Try it.

```
v = TimeSeries([ random() for i in [1..100] ])
```

```
v.plot(points=True)
```

```
v.mean()
```

```
0.50932292235223053
```

```
v.variance()
```

```
0.094766349242779893
```

Problem 5: Interact

You can make interactive controls in Sage. Here is an example:

```
@interact
def f(n=2,m=(1..100)):
   print "n * m = ", n*m
```

n 2

m

```
n * m =  2
```

CPU time: 0.00 s, Wall time: 0.00 s

Problem: Make an interact with three sliders that multiplies the three numbers given by the sliders.

Problem: Add another ticker symbol to this interact for some company you find interesting.

```
symbols = ['bsc', 'vmw', 'sbux', 'aapl', 'amzn', 'goog', 'wfmi',
'msft', 'yhoo', 'ebay', 'java', 'rht', ]; symbols.sort()
stocks = dict([(s, finance.Stock(s)) for s in symbols])

@interact
def data(symbol = symbols, days=(20,(1..400)), alpha=
(0.3,(0,1.0)), beta=(0.5,(0,1.0))):
    S = stocks[symbol]
    html('<h1 align=center>Last %s Days of <font
color="darkred">%s</font></h1>'%(days, S))
    c = S.close()[-days:]
    m = c.exponential_moving_average(alpha)[1:]
    m2 = c.exponential_moving_average(beta)[1:]
    show(c.plot() + m.plot(rgbcolor='#999999') +
m2.plot(rgbcolor='#333333'), figsize=[8,4])
```

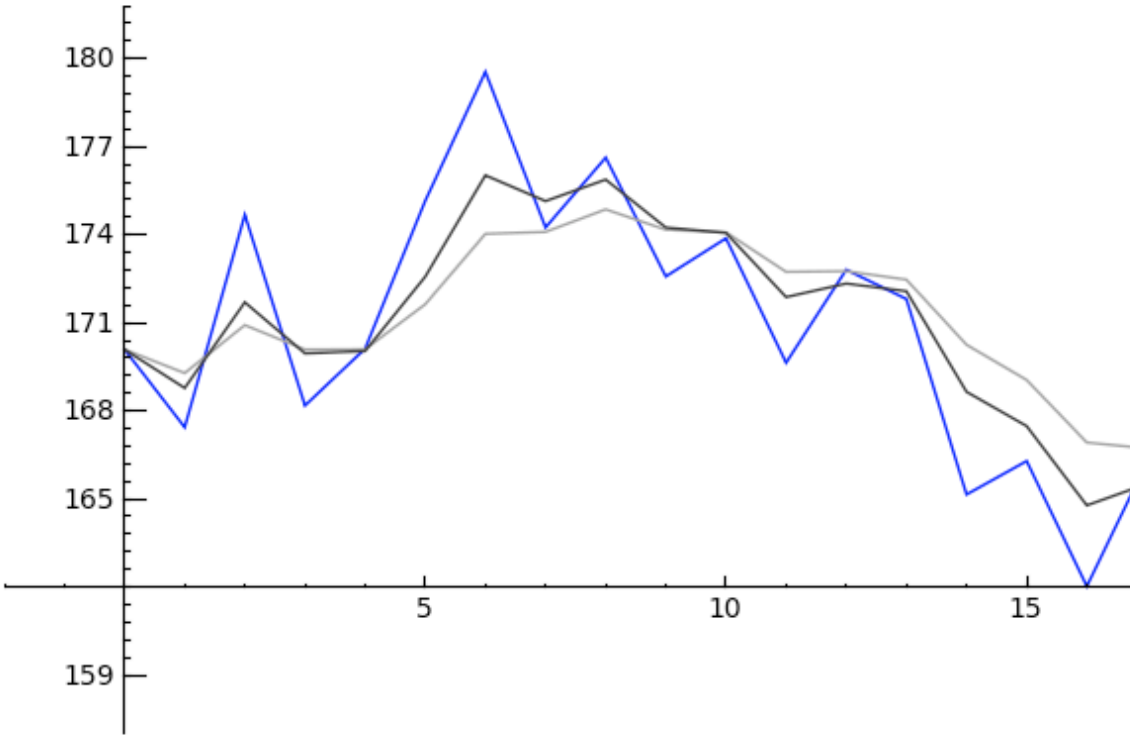
symbol aapl

days

alpha

beta

Last 20 Days of AAPL (155.51)



CPU time: 0.02 s, Wall time: 0.02 s

Problem 7: Investing in Apple this year

Below we download the closing price series for Apple for this year then see how much money one would make by buying and holding between January 1 and July 28, and compare this to

how much one would make by correctly predicting whether Apple stock would go up or down each day and by how much.

Scenario 1: Buy and Hold

```
# download last 143 closing prices for apple, which today means
everything since jan 2.
v = finance.Stock('aapl').close()[-143:]
```

```
# plot it
v.plot()
```

```
v[0]  # price on Jan 2
```

```
194.84
```

```
v[-1] # -1 = last entyr -- price today
```

```
162.12
```

```
# SO, how much do you "profit" with buy and hold?
```

Scenario 2: You're a time traveler :-)

Now imagine you are a time traveler from the future and you know the prices of Aapl every day. You thus stay buy when it

will go up and short when it will go down. By what factor can you increase your investment by the end of the year?

```
# download last 143 closing prices for apple, which today means
everything since jan 2.
v = finance.Stock('aapl').close()[-143:]
```

```
# HINT: v.diffs() gives the differences.
print v.diffs()
print v.abs() # gives the absolute values
print v.sum() # sums up the values in v
```

```
[0.0900, -14.8800, -2.4100, -6.3900, 8.1500 ... 1.1400, -4.2700,
4.2400, -7.2300, 3.0900]
[194.8400, 194.9300, 180.0500, 177.6400, 171.2500 ... 166.2900,
162.0200, 166.2600, 159.0300, 162.1200]
22655.89
```