Akshay Vijayendiran
Alyssa Liu
Zhehao Li

Assignment 1 Solution

IEOR E4729
Ken Gleason
April 16, 2020

1. (2 point) Tatonement is:

   a) a special form of continuous trading

   b) **an iterative series of auctions**

   c) an example of a dealer market

2. (2 point) The largest components of transaction costs are:

   a) Commissions, Fees and Taxes

   b) Spreads and Price Appreciation

   c) **Market Impact, Timing Risk and Opportunity Cost**

3. (6 points) Given the following order book, and new orders, show the order state of the order book and any trades that are generated after each order arrives. If a trade does not occur note "no trade". The starting point for each question is the cumulative order book from the previous question.

Bid Quantity	Bid Price	Ask Price	Ask Quantity
2000	22.20	22.21	1200
1800	22.19	22.25	2300
1000	22.16	22.26	1000
1200	22.15	22.30	2000

   (a) A new order arrives to Buy 1000 shares at $22.20.

Bid Quantity	Bid Price	Ask Price	Ask Quantity
3000	22.20	22.21	1200
1800	22.19	22.25	2300
1000	22.16	22.26	1000
1200	22.15	22.30	2000

   (b) A new order arrives to Sell 1200 shares at $22.19.

Bid Quantity	Bid Price	Ask Price	Ask Quantity
1800	22.20	22.21	1200
1800	22.19	22.25	2300
1000	22.16	22.26	1000
1200	22.15	22.30	2000

(c) A new order arrives to Buy 2200 shares at $22.24.

Bid Quantity	Bid Price	Ask Price	Ask Quantity
1000	**22.24**	22.25	2300
1800	22.20	22.26	1000
1800	22.19	22.3	2000
1000	22.16		
1200	22.15		

(d) A new order arrives to Sell 2000 shares at $22.19.

Bid Quantity	Bid Price	Ask Price	Ask Quantity
800	22.20	22.25	2300
1800	22.19	22.26	1000
1000	22.16	22.30	2000
1200	22.15		

(e) A market order order arrives to buy 3000 shares.

Bid Quantity	Bid Price	Ask Price	Ask Quantity
800	22.20	22.26	**300**
1800	22.19	22.30	2000
1000	22.16		
1200	22.15		

4. (4 points) Using atvi.csv from the data directory, load the data using pandas and selecting the data for September 27, 2019 **only**:

a) Calculate the difference between the Open and Close prices in each interval

```
# Loading tick data
tickfile = 'atvi'
data = pd.read_pickle(tickfile)

# Picking the specific day
data = data.loc[:,'2019-09-27'].copy()

# Computing the difference of open and close
# Either is fine (open-close) or (close-open) in grading
data.loc[:,'diff'] = data.loc[:,'close'] - data.loc[:,'open']
```

b) Calculate the return between each Open and the previous Open.

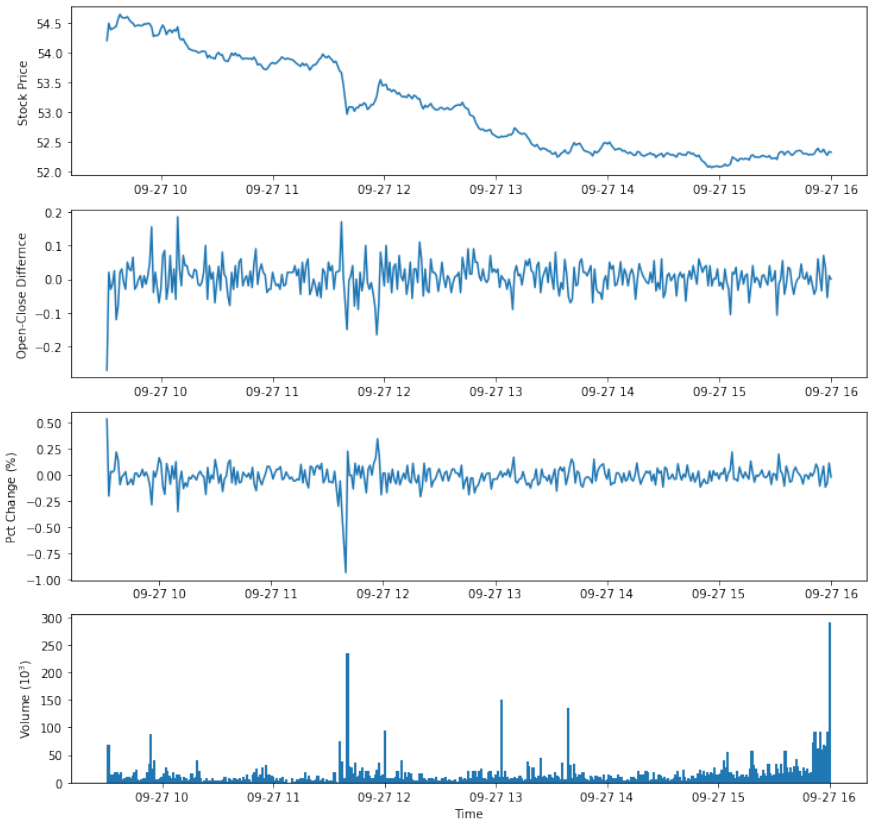
```
# Computing the percentt change of open price
data.loc[:,'pct_change'] = data.loc[:,'open'].pct_change()
```

timestamp	open	high	low	close	volume	diff	pct_change
2019-09-27 09:31:00	54.2100	54.5000	54.210	54.4800	69539	0.270	NaN
2019-09-27 09:32:00	54.5000	54.5500	54.440	54.4800	10478	-0.020	0.005350
2019-09-27 09:33:00	54.3909	54.5272	54.327	54.4209	13645	0.030	-0.002002
2019-09-27 09:34:00	54.4100	54.4500	54.370	54.4250	9405	0.015	0.000351
2019-09-27 09:35:00	54.4250	54.5000	54.370	54.4000	19844	-0.025	0.000276
...	...	...	...	...	...	...	...
2019-09-27 15:56:00	52.3800	52.3850	52.290	52.3100	59610	-0.070	0.000860
2019-09-27 15:57:00	52.3200	52.3300	52.280	52.2850	68304	-0.035	-0.001145
2019-09-27 15:58:00	52.2800	52.3400	52.280	52.3350	66949	0.055	-0.000765
2019-09-27 15:59:00	52.3400	52.3500	52.330	52.3300	93433	-0.010	0.001148
2019-09-27 16:00:00	52.3300	52.3600	52.280	52.3300	291624	0.000	-0.000191

387 rows × 7 columns

c) Create a chart with four rows of plots: stock price, close - open, percent change, volume per interval (as bar).

```
fig, axs = plt.subplots(4, 1, figsize = (12, 12))

axs[0].plot(taq_1.index, taq_1.loc[:,'open'])
axs[0].set_ylabel('Stock Price')

axs[1].plot(taq_1.index, taq_1.loc[:,'diff'])
axs[1].set_ylabel('Open-Close Differnce')

axs[2].plot(taq_1.index, 100 * taq_1.loc[:,'pct_change'])
axs[2].set_ylabel('Pct Change ($\%$)')

axs[3].bar(taq_1.index, taq_1.loc[:,'volume'] / 1000, width = 0.001)
axs[3].set_ylabel('Volume ($10^3$)')
axs[3].set_xlabel('Time')

plt.show()
```



5. (6 points) Using, atvi.csv from the data directory, load the data using pandas:

   (a) Using the example code from the class or your own method fit a function to obtain coefficients for a VWAP target function. Show your resulting coefficients.

1 Loading Data

```
bars = pd.read_pickle("atvi.zip")
bars = bars.sort_index(ascending=True )
```

2 Caculating the Accumlative Quantity

```
# calculate the cumulative pct by day
totl_volume = bars.groupby( [ bars.index.date ] ).tail( 1 )[ 'volume' ]
bars[ 'accum_volume' ] = bars.groupby( [ bars.index.date ] ).cumsum()[ 'volume' ]
bars[ 'accum_pct' ] = bars.groupby( [ bars.index.date ] )[ 'accum_volume' ].transform( lambda x: x / x.iloc[ -1

# add a minute bin
# US start of date, calc in minutes
start_of_day = (9 * 60) + 30
# get the time for each bin in minutes and subtract 9:30
bars[ 'minute_bars' ] = (bars.index.hour * 60) + bars.index.minute - start_of_day
```

3 Linear Regression - Question (a)

```
# arrange our data
minute_bars = bars[ 'minute_bars' ]
X = pd.DataFrame( { 'bin': minute_bars,
                    'bin2' : minute_bars**2,
                    'bin3' : minute_bars**3,
                    'bin4' : minute_bars**4,
                    'bin5' : minute_bars**5} )
y = bars[ 'accum_pct' ]
```

```
# now do the regression with no intercept
lm = linear_model.LinearRegression( fit_intercept = False )
model = lm.fit( X, y )
predictions = lm.predict( X )

# Rsquared
lm.score( X, y )
```

```
0.8036359282074419
```

```
print ('The coefficients of the function are: ')
print(lm.coef_)
```

```
The coefficients of the function are:
[ 5.37368693e-03 -2.91686024e-05  7.21057734e-08 -2.51583797e-11
 -4.66490653e-14]
```

(b) Run the VWAP target function and plot your resulting cumulative volume function/target curve.

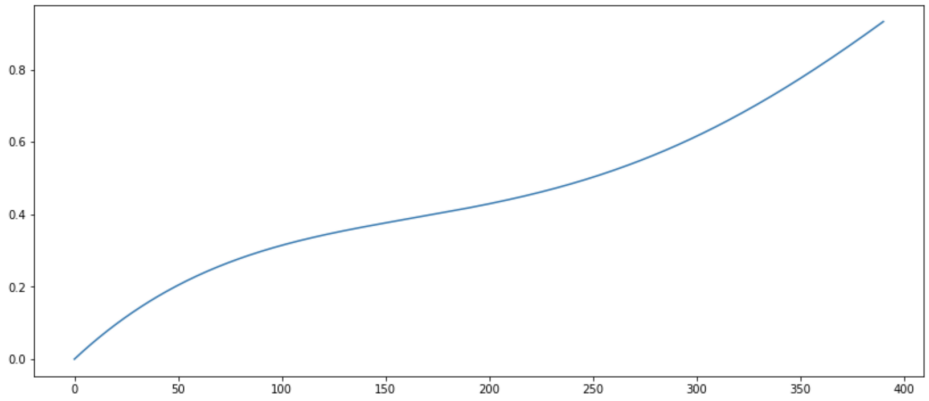
4 VWAP functions Visualisation (Question (b))

```
def vwap_target( bar_num, coefs ):
    return ( coefs[ 0 ] * bar_num +
             coefs[ 1 ] * bar_num**2 +
             coefs[ 2 ] * bar_num**3 +
             coefs[ 3 ] * bar_num**4 +
             coefs[ 4 ] * bar_num**5 )
```

```
bins = np.arange(0,391)
tgts = vwap_target( bins, lm.coef_ )
```

```
plt.plot(tgts)
```

```
[<matplotlib.lines.Line2D at 0x1a21a33550>]
```



(c) Plot the same VWAP target function as in b) above but also plot the "close" series from September 27, 2019 on the same chart.

5 Linear Regression Fit and Close Price Visulaization (Question (c))

```
one_day = bars[ '9-27-2019' ][ 'accum_pct' ]
close = bars[ '9-27-2019' ]['close']
tgts_day = vwap_target( bars[ '9-27-2019' ]['minute_bars'], lm.coef_ )
```

```
fig, ax1 = plt.subplots()

color2 = 'tab:orange'
ax1.set_ylabel('accum_pct')
ax1.plot( tgts_day, label = 'Linear Regression Fit' ,color='m')
ax1.plot( one_day, label = '9-27-2019' ,color=color2)
ax1.tick_params(axis='y')
plt.legend(loc=6)

ax2 = ax1.twinx()               # instantiate a second axes that shares the same x-axis
ax2.set_ylabel('close')         # we already handled the x-label with ax1
ax2.plot(close,label = 'close')
ax2.tick_params(axis='y')

fig.tight_layout()              # otherwise the right y-label is slightly clipped
plt.title( 'VWAP Profile Fitting and a Single Day with "close"' )
plt.legend(loc=4)
plt.show()
```