Homework 9 - Berkeley STAT 157

In [1]:
```
import d2l
import math
import mxnet as mx
from mxnet import autograd, gluon, init, nd
from mxnet.gluon import loss as gloss, nn, rnn
from mxnet.gluon import data as gdata
import time
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
import numpy as np
from scipy.stats import norm
from sklearn.preprocessing import StandardScaler
from scipy import stats
import warnings
warnings.filterwarnings('ignore')
%matplotlib inline
```

1. Time Series Model

1.1

Generally speaking, log-scale informs on relative changes (multiplicative), while linear-scale informs on absolute changes (additive). And we care about the relative change in the analysis of this question. The log price could ensure that equivalent price changes could be represented by the same vertical distance on the scale. This explains why we take log for the open and close price. Similarly, it also explains why we take logarithm on volumes and the log volume tells the liquidity of the security. In the Random Walk Hypothesis, log of the prices follows random walks. This also stays in line with the Black-Scholes model where the changes of prices follow the lognormal distribution and they are independently and identically distributed.

For high and low prices, we take log of the ratio of them over the open price to connect these prices with open price, whose distribution could hence be linked by this bijective transformation.

1.2

By rescaling with 10, the prices changes will not be too small to be included in full length in the calculation. This would also maintain the centralisation of 0.

1.3

For Guassian models, given the predicted values $\hat{z}_{st}$, the variable z_{st} and a variance σ^2 (a constant), the Guassian model density function $f(x)$ for the loss would be:

$$f(z_{st};\hat{z}_{st},\sigma^2) = \frac{1}{2\pi\sigma^2}\exp(-\frac{(\hat{z}_{st}-z_{st})^2}{\sigma^2})$$

By taking the logorithm,

$$\log f(z_{st};\hat{z}_{st},\sigma^2) = -\frac{1}{2}(log(2\pi\sigma^2)+\frac{(\hat{z}_{st}-z_{st})^2}{\sigma^2}) \propto (\hat{z}_{st}-z_{st})^2$$

Hence the log-normal density function is just the square loss rescaled by $-\frac{1}{2\sigma^2}$ and transformed by a function of σ^2. Therefore the prediction error follows a log-normal distribution.

1.4

This could be becasue the covariance and correlation among these stocks (which are not independent of one another) might not be changing much over time. This would result in multivariate distribution of the prices changes when estimating jointly.

Some illustration of Problem 1

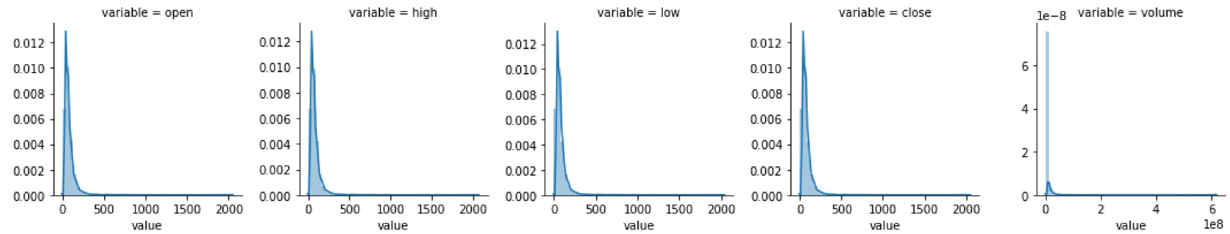
In [2]:
```
df = pd.read_csv('all_stocks_5yr.csv')
df.describe()
```

Out[2]:

	open	high	low	close	volume
count	619029.000000	619032.000000	619032.000000	619040.000000	6.190400e+05
mean	83.023334	83.778311	82.256096	83.043763	4.321823e+06
std	97.378769	98.207519	96.507421	97.389748	8.693610e+06
min	1.620000	1.690000	1.500000	1.590000	0.000000e+00
25%	40.220000	40.620000	39.830000	40.245000	1.070320e+06
50%	62.590000	63.150000	62.020000	62.620000	2.082094e+06
75%	94.370000	95.180000	93.540000	94.410000	4.284509e+06
max	2044.000000	2067.990000	2035.110000	2049.000000	6.182376e+08

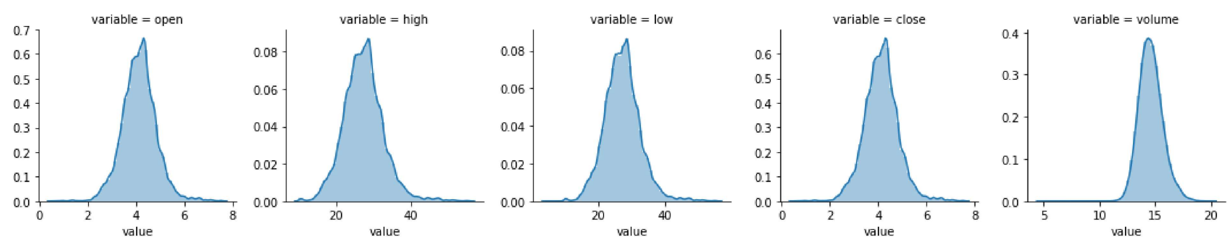
Example of Logrithm

In [36]:
```python
df1 = df.copy()
quantitative = [f for f in df1.columns[1:6]]
f = pd.melt(df1, value_vars=quantitative)
g = sns.FacetGrid(f, col="variable", col_wrap=5, sharex=False, sharey=False)
g = g.map(sns.distplot, "value")
plt.show()
```

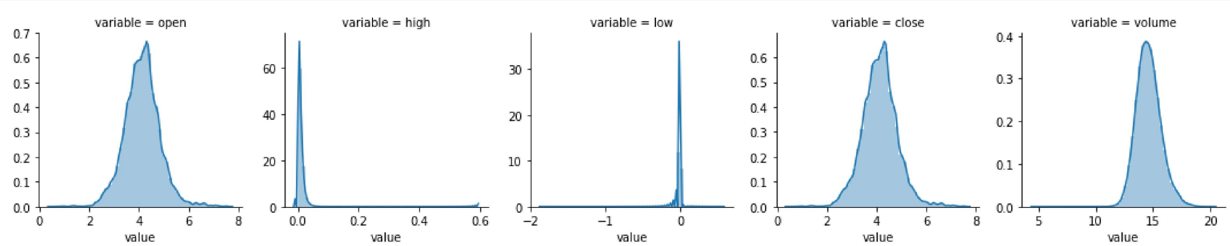


In [51]:
```python
df2 = df1.copy()
df2['open'] = np.log(df1['open'])
df2['high'] = 10* (np.log(df1['high']) - np.log(df1['open']))
df2['low'] = 10* (np.log(df1['low']) - np.log(df1['open']))
df2['close'] = np.log(df1['close'])
df2['volume'] = np.log(df1['volume'])
```

In [45]:
```python
# with rescale 10
quantitative = [f for f in df2.columns[1:6]]
f = pd.melt(df1, value_vars=quantitative)
g = sns.FacetGrid(f, col="variable", col_wrap=5, sharex=False, sharey=False)
g = g.map(sns.distplot, "value")
plt.show()
```



In [52]:
```python
#without rescale
quantitative = [f for f in df2.columns[1:6]]
f = pd.melt(df2, value_vars=quantitative)
g = sns.FacetGrid(f, col="variable", col_wrap=5, sharex=False, sharey=False)
g = g.map(sns.distplot, "value")
plt.show()
```



\pagebreak