CryptoCurrency

March 30, 2020

In [7]:
```
import numpy as np
import pandas as pd
#change the path here if necessary
data_Bitmex = pd.read_csv("/Users/jizngziwei/Downloads/Bitmex/Depth_20200101_btc_usd.cs
data_okex = pd.read_csv("/Users/jizngziwei/Downloads/okex/Depth_20200101_btc_usd.csv")
```

In [8]:
```
import time
import datetime
from datetime import date
def timestampProcess(data):
    data["DateTime"] = pd.Series(list(map(lambda x: time.strptime(x, "%Y-%m-%d %H:%M:%S
    return data
```

In [9]:
```
#if you are the first time run this notebook, run code below
data_Bitmex = timestampProcess(data_Bitmex)
data_okex = timestampProcess(data_okex)
```

In [10]:
```
import re
def getBidsPrice(data):
    pattern = r'BidsPrice'
    index_list = ["DateTime"]
    for idx in list(data.index):
        if re.search(pattern, idx):
            index_list.append(idx)
    return data[index_list]
def getBidsQuantity(data):
    pattern = r'BidsQuantity'
    index_list = ["DateTime"]
    for idx in list(data.index):
        if re.search(pattern, idx):
            index_list.append(idx)
    return data[index_list]
def getAsksPrice(data):
    pattern = r'AsksPrice'
    index_list = ["DateTime"]
    for idx in list(data.index):
        if re.search(pattern, idx):
            index_list.append(idx)
```

```
            return data[index_list]
        def getAsksQuantity(data):
            pattern = r'AsksQuantity'
            index_list = ["DateTime"]
            for idx in list(data.index):
                if re.search(pattern, idx):
                    index_list.append(idx)
            return data[index_list]
        def getAvgPrice(data):
            return data.iloc[1:].mean()
```

In [25]:
```
def profitPerTrans(data1, data2, ticker, budget):
    #ticker= True: data1_bid>data2_ask, Buy index2 sell to index1
    if ticker == True:
        BidsPrice = getBidsPrice(data1)
        BidsQuantity = getBidsQuantity(data1)
        AsksPrice = getAsksPrice(data2)
        AsksQuantity = getAsksQuantity(data2)
    elif ticker == False:
        BidsPrice = getBidsPrice(data2)
        BidsQuantity = getBidsQuantity(data2)
        AsksPrice = getAsksPrice(data1)
        AsksQuantity = getAsksQuantity(data1)
    else:
       # print("No profitable chance.")
        return
    if BidsPrice[0] <= AsksPrice[0]:
        #you can change time interval limit here
       #     print("Unable to make transaction because of time interval problem.")
            return
    current_position = 0
    TransQuantity = []
    PriceDiff = []
    for i in range(1,min(len(BidsQuantity),len(AsksQuantity))):
        if current_position < budget:
            TransQuantity.append(min(BidsQuantity[i], AsksQuantity[i]))
            current_position = np.dot(np.array(TransQuantity), AsksPrice[1:i+1])
            PriceDiff.append(BidsPrice[i] - AsksPrice[i])
        else:
            continue
    return np.dot(TransQuantity, PriceDiff)
```

In [12]:
```
def getTicker(data1, data2):
     #ticker= True: data1_bid>data2_ask, Buy index2 sell to index1
    ticker = None
    if getAvgPrice(getBidsPrice(data1))> getAvgPrice(getAsksPrice(data2)):
        ticker = True
    elif getAvgPrice(getBidsPrice(data2)) > getAvgPrice(getAsksPrice(data1)):
```

```
                ticker = False
            return ticker
```

```
In [80]: data1 = data_Bitmex.iloc[2172,:]
         data2 = data_okex.iloc[3358,:]

         profitPerTrans(data1, data2, getTicker(data1, data2), 100000)
```

```
Out[80]: 12374.699999998846
```

```
In [90]: def sortedDictValues1(adict):
             new_dict = {}
             for key in sorted(adict.keys()):
                 new_dict[key] = adict[key]
             return new_dict
```

```
In [91]: import random
         from random import *

         def generateIntervalSeries(start,end,k):
             s = sorted(sample(range(start,end),k))
             result = {}
             for i in range(len(s)-1):
                 for j in range(i,len(s)):

                     data1 = data_Bitmex.iloc[s[i],:]
                     data2 = data_okex.iloc[s[j],:]

                     if profitPerTrans(data1, data2, getTicker(data1, data2), 100000)==None:
                         j=j+1
                     elif profitPerTrans(data1, data2, getTicker(data1, data2), 100000) > 100 a
                         #you can change the limit for comission here
                         result[s[i]] = s[j]
                         i=j
                         j=i+1
                     else:
                         j=j+1
             return sortedDictValues1(result)
```

```
In [123]: intervalSeries = generateIntervalSeries(0,len(data_Bitmex),200)
```

```
In [132]: def grossProfit(intervalSeries):
              profit = 0
              for i in intervalSeries.keys():
                  j = intervalSeries[i]
                  data1 = data_Bitmex.iloc[i,:]
                  data2 = data_okex.iloc[j,:]
                  profit = profit + profitPerTrans(data1, data2, getTicker(data1, data2), 1000
              return profit
```

```
In [133]: grossProfit(intervalSeries)
```

```
Out[133]: 3903143.699999981
```