Blockchain Technology and Data Economics

Practice 4 - Exploring bitcoin transaction graphs

Version: 2020 Spring

Introduction

In this practise, we will learn about how to get the blockchain data, and provide step-by-step information as to how to explore, clean up, analyse, and visualize this data.

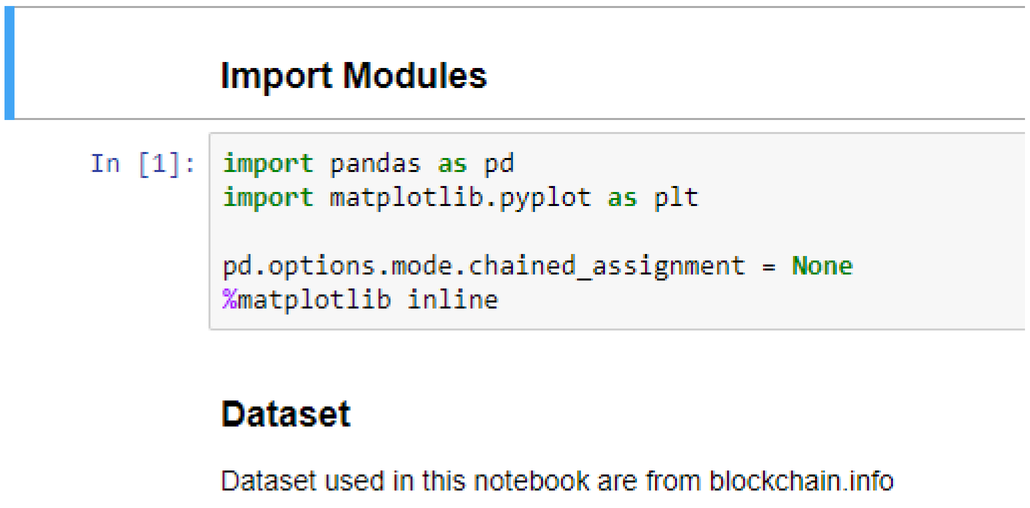
I. Bitcoin and blockchain graphs

Blockchain.info is one of the best places to look at the latest bitcoin stats and graphs. There are different kinds of charts and graphs concerning bitcoin and blockchain that are available for analysis. We can also download the data in a variety of formats—CSV, JSON, and so on. We have downloaded some of this data in CSV format in the previous section, and now we will explore this data in a Jupyter Notebook.

We start by importing the modules we need. We need pandas for data reading, exploration, and cleanup, and we need *matplotlib* for creating the graphs.

Look at the data showing the total number of bitcoins in circulation. Read the CSV file that has this data and create a *pandas DataFrame*.

The following screenshot shows the data for the total number of bitcoins in circulation:



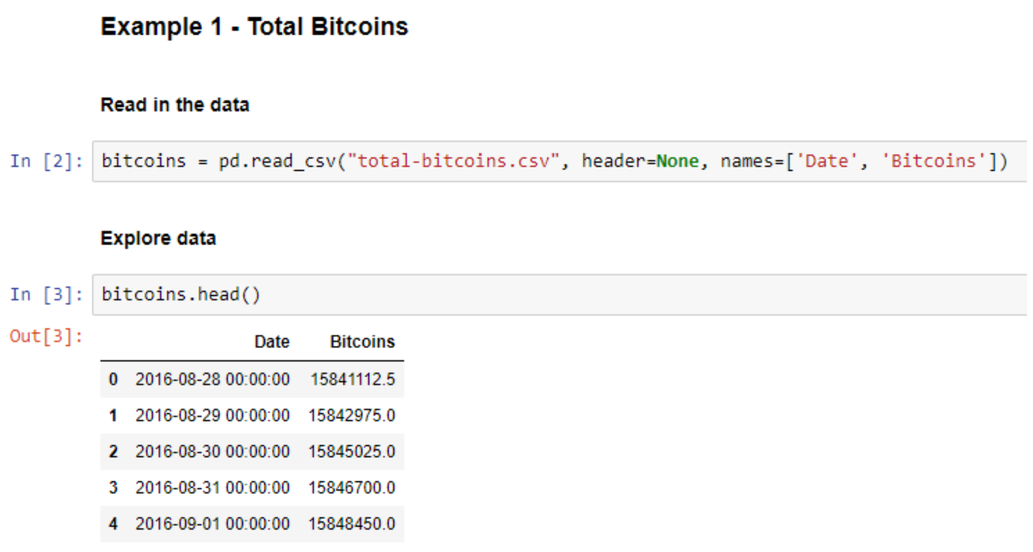
Import Modules

```
In [1]: import pandas as pd
        import matplotlib.pyplot as plt

        pd.options.mode.chained_assignment = None
        %matplotlib inline
```

Dataset

Dataset used in this notebook are from blockchain.info



Example 1 - Total Bitcoins

Read in the data

```
In [2]: bitcoins = pd.read_csv("total-bitcoins.csv", header=None, names=['Date', 'Bitcoins'])
```

Explore data

```
In [3]: bitcoins.head()
```

Out[3]:

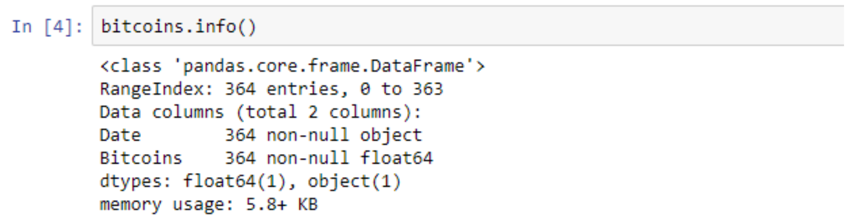
	Date	Bitcoins
0	2016-08-28 00:00:00	15841112.5
1	2016-08-29 00:00:00	15842975.0
2	2016-08-30 00:00:00	15845025.0
3	2016-08-31 00:00:00	15846700.0
4	2016-09-01 00:00:00	15848450.0

II. Exploring, cleaning up, and analyzing data

In order to explore this data, we use the *head()* method to look at the records from the top and we call the *info()* method on the DataFrame to get some more information, such as how many records there are, how many null or missing records there are, or what the data types of the various columns are.

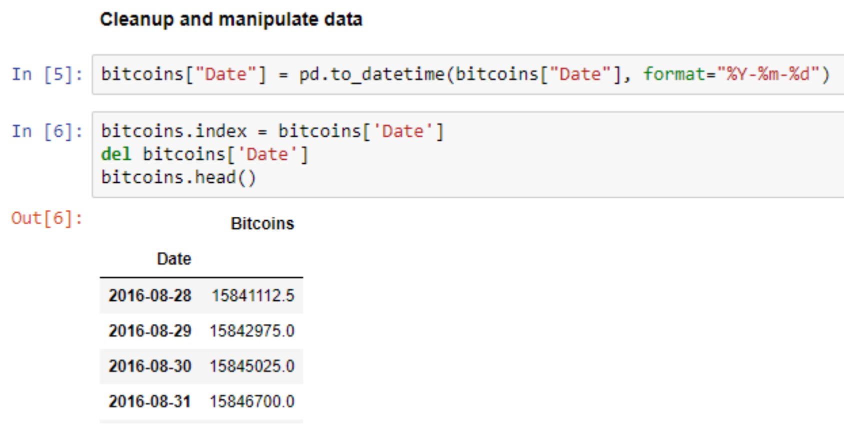
We can see that the Date column is shown as object. We change this to date-time to visualize this data. To do this, we use the *to_datetime* method, and assign the converted values back to the same column—the DataFrame.

The following screenshot depicts the date format of bitcoins:



```
In [4]: bitcoins.info()

<class 'pandas.core.frame.DataFrame'>
RangeIndex: 364 entries, 0 to 363
Data columns (total 2 columns):
Date        364 non-null object
Bitcoins    364 non-null float64
dtypes: float64(1), object(1)
memory usage: 5.8+ KB
```



Cleanup and manipulate data

```
In [5]: bitcoins["Date"] = pd.to_datetime(bitcoins["Date"], format="%Y-%m-%d")
```

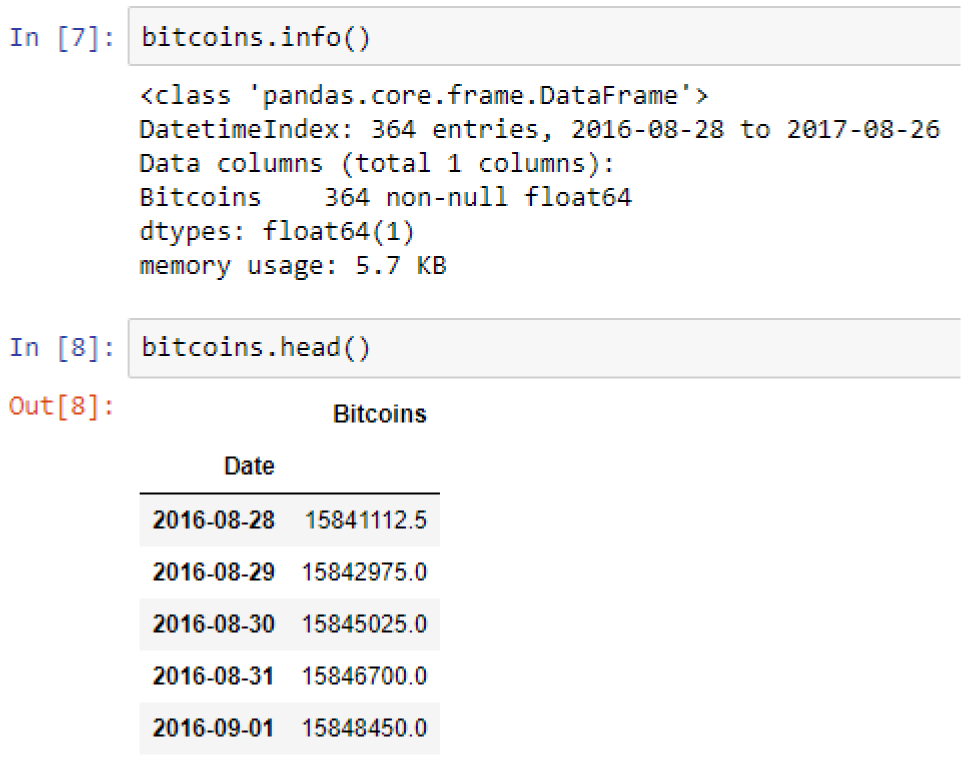
```
In [6]: bitcoins.index = bitcoins['Date']
        del bitcoins['Date']
        bitcoins.head()
```

Out[6]:

	Bitcoins
Date	
2016-08-28	15841112.5
2016-08-29	15842975.0
2016-08-30	15845025.0
2016-08-31	15846700.0

Set the index of the DataFrame to the Date column and delete the Date column as a separate column. Perform this step in order to take advantage of the time series features of pandas.

Now, check again whether the changes took place by calling info and head on the DataFrame.
The following screenshot shows the bitcoins for a particular range of dates:



```
In [7]: bitcoins.info()

<class 'pandas.core.frame.DataFrame'>
DatetimeIndex: 364 entries, 2016-08-28 to 2017-08-26
Data columns (total 1 columns):
Bitcoins    364 non-null float64
dtypes: float64(1)
memory usage: 5.7 KB
```

```
In [8]: bitcoins.head()
```

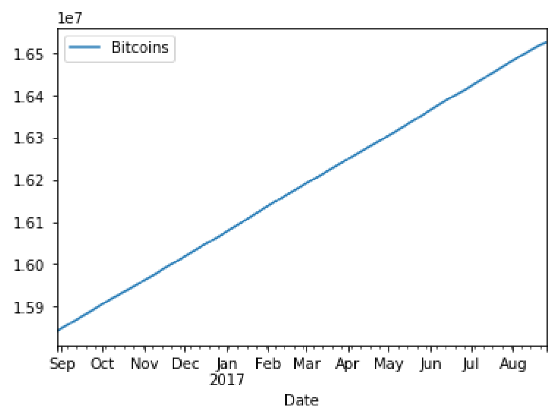
Out[8]:

	Bitcoins
Date	
2016-08-28	15841112.5
2016-08-29	15842975.0
2016-08-30	15845025.0
2016-08-31	15846700.0
2016-09-01	15848450.0

We are now ready to create a graph from this data. Call the *plot ()* method on the DataFrame and then call the *show ()* method to display the graph.
It shows the total number of bitcoins that have already been mined over the time period for which we have this record.
The following screenshot describes the graph for the preceding data:

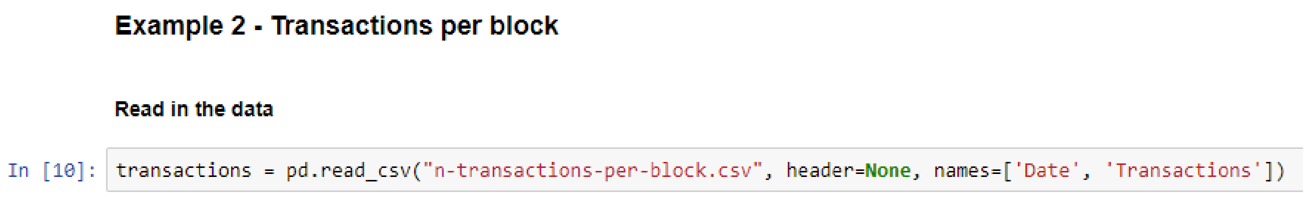
Visualize Data

```
In [9]: bitcoins.plot()
        plt.show()
```



III. Visualizing data

Let's look at another example. Here, we are looking at transactions for block data that we read into the data:

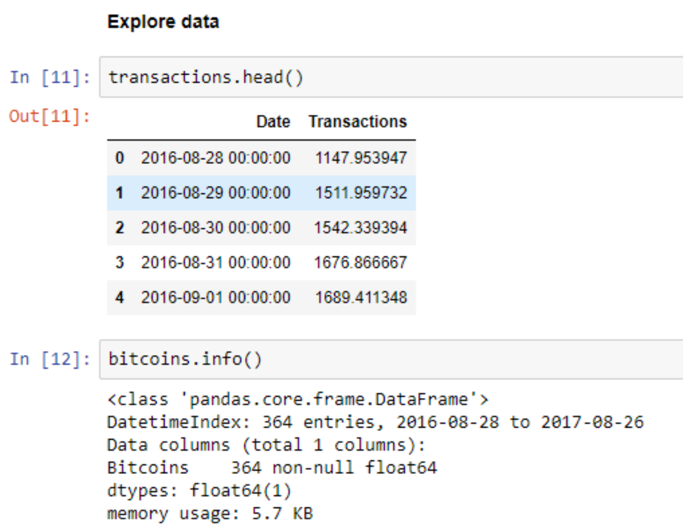


Example 2 - Transactions per block

Read in the data

```
In [10]: transactions = pd.read_csv("n-transactions-per-block.csv", header=None, names=['Date', 'Transactions'])
```

Initially, we visually explore this data using the head and info methods, as shown in the following screenshot:



Explore data

```
In [11]: transactions.head()
```

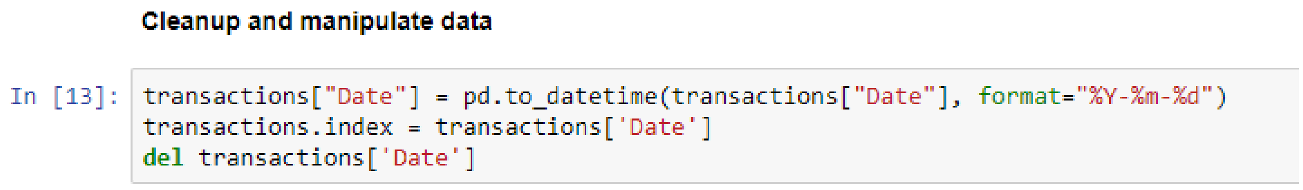
Out[11]:

	Date	Transactions
0	2016-08-28 00:00:00	1147.953947
1	2016-08-29 00:00:00	1511.959732
2	2016-08-30 00:00:00	1542.339394
3	2016-08-31 00:00:00	1676.866667
4	2016-09-01 00:00:00	1689.411348

```
In [12]: bitcoins.info()

<class 'pandas.core.frame.DataFrame'>
DatetimeIndex: 364 entries, 2016-08-28 to 2017-08-26
Data columns (total 1 columns):
Bitcoins    364 non-null float64
dtypes: float64(1)
memory usage: 5.7 KB
```

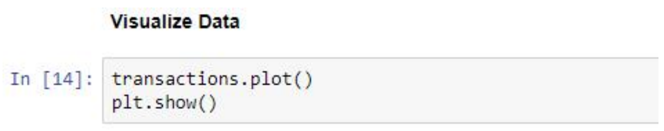
Next, we clean up, convert, and reshape the data, as shown in the following screenshot:



Cleanup and manipulate data

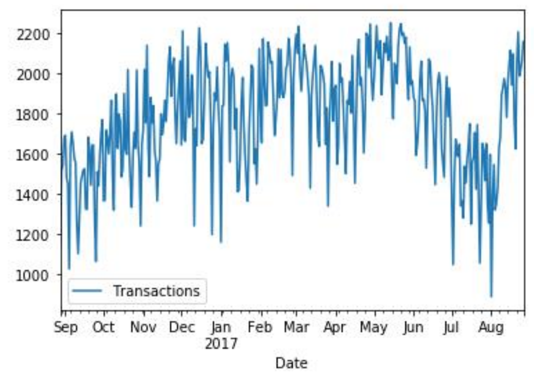
```
In [13]: transactions["Date"] = pd.to_datetime(transactions["Date"], format="%Y-%m-%d")
         transactions.index = transactions['Date']
         del transactions['Date']
```

Finally, we visualize our transactions in the block data, as shown in the following screenshot:



Visualize Data

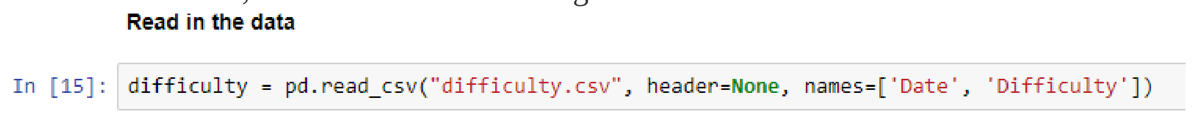
```
In [14]: transactions.plot()
         plt.show()
```



IV. Mining Difficulty

Similarly, there is another example that we should look at regarding the data showing mining difficulty. The steps for mining data are as follows:

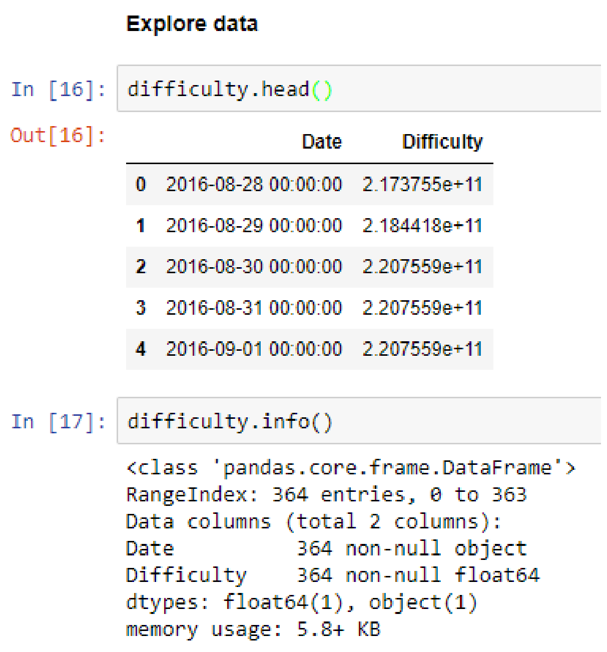
1. Read in the data, as shown in the following screenshot:



Read in the data

```
In [15]: difficulty = pd.read_csv("difficulty.csv", header=None, names=['Date', 'Difficulty'])
```

2. Explore the data, as shown in the following screenshot:



Explore data

```
In [16]: difficulty.head()
```

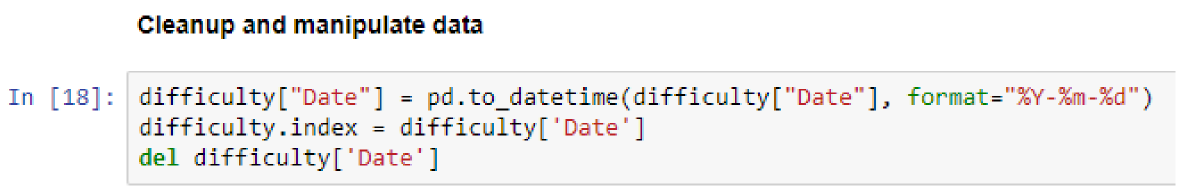
Out[16]:

	Date	Difficulty
0	2016-08-28 00:00:00	2.173755e+11
1	2016-08-29 00:00:00	2.184418e+11
2	2016-08-30 00:00:00	2.207559e+11
3	2016-08-31 00:00:00	2.207559e+11
4	2016-09-01 00:00:00	2.207559e+11

```
In [17]: difficulty.info()

<class 'pandas.core.frame.DataFrame'>
RangeIndex: 364 entries, 0 to 363
Data columns (total 2 columns):
Date          364 non-null object
Difficulty    364 non-null float64
dtypes: float64(1), object(1)
memory usage: 5.8+ KB
```

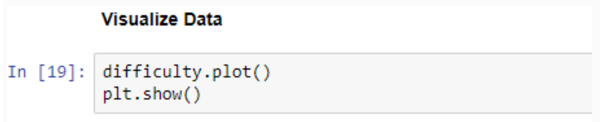
3. Clean up the data, as shown in the following screenshot:



Cleanup and manipulate data

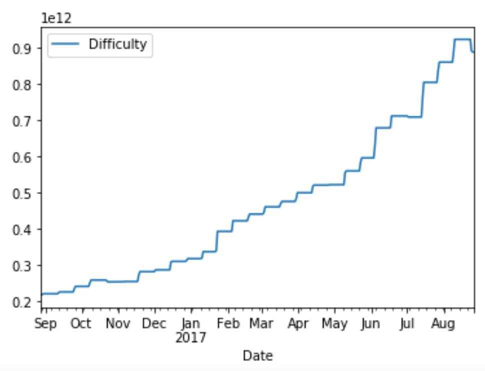
```
In [18]: difficulty["Date"] = pd.to_datetime(difficulty["Date"], format="%Y-%m-%d")
         difficulty.index = difficulty['Date']
         del difficulty['Date']
```

4. Finally, visualize the data, as shown in the following screenshot:



Visualize Data

```
In [19]: difficulty.plot()
         plt.show()
```



We can see that there has been a gradual increase in mining difficulty over the years, and it trends upwards. These are just a few examples of transaction graphs. There is a lot of other data available for you to explore from the bitcoin and blockchain ecosystem.

Well done!! Show your results to TA!

The End of Practice