ETL Technical Report:  
Bitcoin & Bitcoin Cash

Rutgers Data Science Bootcamp

January 2020

Christos Ferderigos, Zohna Jones, Omar Kreidieh, Anu Srivastava

# 

# Purpose

Bitcoin and Bitcoin Cash are two popular and increasingly profitable cryptocurrencies on the market. Although Bitcoin Cash is a fork of Bitcoin it’s gaining users and user value, making it comparable for analysis. To demonstrate our understanding of ETL (extract, transform load), we examined Bitcoin and Bitcoin Cash data. The purpose of performing the ETL process on the data was to analyze the relationship between Bitcoin and Bitcoin cash for a year time span. Our goal was to determine whether the two currencies have any relationship.

# Extract

Two data sources of different formats were used for research. The Bitcoin Cash price dataset was a .csv from a Kaggle source, which included date, open, high, low, close, volume, and market cap columns on a daily basis from 7/23/17 to 2/20/18. Bitcoin data was called from the CoinGecko API/JSON. A loop was used to call dates for 2017 through 2019 in day-month-year format and a dataframe was created to hold the results that included date, market cap, volume, and current price columns. Bitcoin data was a little more challenging to extract because a loop needed to be created to generate date variables to then get requests for the series of dates. A try-except was also built in to the loop to accommodate for varying number of days per month. Bitcoin data had to be narrowed down to match the Bitcoin Cash dates for the first layer of data cleaning.

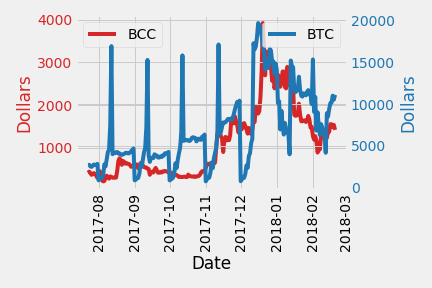
# Transform

The datasets were cleaned using Python Pandas through a Jupyter Notebook. The date on the Bitcoin API/JSON dataset had to be reformatted to match the year-month-day Bitcoin Cash date format and inner merged on date. Date columns had to be dropped and renamed to align the two datasets and to assure terminology was kept consistent for column names. Rows had to be removed that contained empty fields in the market cap column, which was a bit challenging because these fields were referenced with (-) instead of Na/NaN. The cleaned Bitcoin and Bitcoin Cash dataframes were combined and a second dataframe with an index variable column was created, which makes it easier for the user to reference performance. Since data will be loaded into a Pg4admin SQL database, it was easier and faster to search using index variables versus searching through a string.

# Load

Pg4Admin SQL was the database used to load the data because the two datasets were relational and the data was being joined on one element, date. Two tables were created, one for index, as the primary key and one for the combined data, which references the primary key to link the two tables. We decided to separate out the label names to improve performance of the database for any large queries. To further normalize the data, a line graph was created using Matplotlib to show the trends of Bitcoin and Bitcoin cash for common dates.

# Analysis

****

From the graph, we can see the Bitcoin Cash generally follows the trend of Bitcoin. If we were to pull in any other coin, we hypothesize that they would follow a similar trend, as Bitcoin is the “gold” standard of cryptocurrencies.

# Further Investigation

We normalized the data so that any user can clone our repository and add additional dates worth of information or additional coins to increase the scope of the analysis.

# Sources:

## Report Sources:

<https://www.bitdegree.org/tutorials/bitcoin-cash-vs-bitcoin/>

## Data Sources:

<https://www.kaggle.com/sudalairajkumar/cryptocurrencypricehistory>

<https://api.coingecko.com/api/v3/coins/>

Extract

* Extract: your original data sources and how the data was formatted (CSV, JSON, pgAdmin 4, etc).

Transform

* CleTransform: what data cleaning or transformation was required.
  + Drop columns that weren’t aligned in each dataset
  + Drop Nan Values for Market Cap-- little difficult because it was just (-) not Na or NaN
  + Rename: Price→ Bitcoin was current price and Bitcoin Cash was close price had to make the same CLOSE\_PRICE
  + Reformatting Date-- API- 02-02-2020 ; CSV-month(spelled out) -day- year
    - Make it fluid so it would read year-month-day
  + Drop dates that don’t align for second layer of cleaning
  + Add column to refer to index as an integer for performance
    - In sql searching through string is more tedious then having index variable thats faster

Load: the final database, tables/collections, and why this was chosen.

* + 1 tables SQL because we were just joining on date to see how they Bitcoin & Bitcoin Cash move-- together or not; just joining on one element
    - No index originally; make our own
    - Make data relational
    - Inner joined on dates--common
    - Foreign key that references primary key to link 2 tables
  + 2nd table→ primary key
  + Normalizing data
  + Line graph of bitcoin\_cash and bitcoin for aligned dates to see the trend