University of St.Gallen

School of Management, Economics, Law, Social Schiences and International Affairs

The effect of Twitter activity on Bitcoin price

Documentation

Software Engineering for Economists
(7,610,1.00)

Dimitrios Koumnakes - 10-613-370
Severin Kranz - 13-606-355
Joël Sonderegger - 11-495-488
Alen Stepic - 11-475-258
Chi Xu - XX-XXX-XXX

Fall Term 2017

Supervisor
Prof. Dr. Philipp Zahn
Department of Economics

December 28, 2017

Contents

1 Introduction **1**

1.1 Goal of the paper 1

1.2 Methodology 1

1.3 Scope 1

2 Setup **2**

2.1 Requirements to run Python scripts 2

2.2 Stata 2

3 Data Collection **2**

3.1 Tweets Data 2

3.1.1 Python Script 2

3.1.2 Hardware Setup 5

3.2 Bitcoin Price Data 7

3.2.1 Execution 7

3.2.2 Output 7

3.2.3 API: Bitcoinaverage.com 8

4 Data Aggregation and Data Wrangling **9**

4.1 Process 9

4.2 Execution 10

4.3 Output 10

5 Data Analysis **11**

List of Figures

1 Twitter Keys and Access Tokens, based on **twitterinc2017c**. 4

2 Data aggregation and data wrangling process 9

List of Abbreviation

API	Application Programming Interface
ATOKEN	Access Token
ASECRET	Access Token Secret
CKEY	Consumer Key
CSECRET	Consumer Secret
JSON	Java Script Object Notation
RPI	Raspberry Pi
SSH	Secure Shell
BPI	Bitcoin Price Index
BTC	Bitcoin
USD	US Dollar

1 Introduction

In the academic environment accountability and reproducibility is important. However, the publishing process of papers and journals seem to be outdated as reproducability is not evaluated by many journals. (**mccullough2013**) New ways of data collection and data processing exist by using computational economics. The usage of algorithms can increase effectiveness and efficiency. Hence, much lager data sets can be proceeded. However this creates also new problems regarding to traceability and reproducibility. Often cited academic paper exist, where the initial computation is not reproducible (**mccullough2013**). Replicating data or existing results do not provide any new knowledge at all. Nevertheless, the ability to reproduce is key in science and non-reproducable work can not be characterized as science or used as a basis for policy making (**mccullough2013**). This explains why reproduction is of great relevance.

1.1 Goal of the paper

The goal of this documentation is the provision of a description. This description should enable the reader to reproduce the results discussed in the separate paper. Thus, it contains an explanation how the input data have been gathered, stored, aggregated and analysed. In other words, the input data, the model core, the model parameters and the applied math program are explained.

1.2 Methodology

This documentation consists out of four chapters. The first chapter contains a short introduction and provides the reader with an overview about the topic. Furthermore it points out the relevance of documentation. The second chapter discusses the input data. This includes the process of gathering and storing twitter tweets as well as the gathering of the bitcoin price data. The third chapter discusses how the data is aggregated by pointing out the core model and its parameters. Finally the fifth chapter discusses how the analysis has been conducted.

1.3 Scope

The scope of the documentation is the provision of an overview about the different steps which have been conducted to obtain the results in the paper. It does not contain any discussions about the results of the separate paper. It is not a deep description of the code as the code itself as the code is documented separately. Nevertheless, important lines of code are discussed.

2 Setup

2.1 Requirements to run Python scripts

Before you go any further, make sure that at least Python version 3.0.0 available from your command line. You can check this by running:

```
python --version
```

You should get some output like Python 3.7.13. If you do not have at least Python version 3.0.0, please install the latest 3.x version from python.org.
On top you need various packages. We created a the requirements file `requirements.txt`, which contains all required packages to run Python scripts of the project. The list of packages to be installed using pip install like so:

```
pip install -r requirements.txt
```

Now you should be ready to run the Python scripts.

2.2 Stata

For the data analysis, we used the statistical software STATA (version 14.2). When seeking for statistical analysis, regression analysis or data management, STATA is a capable tool. To reproduce our results, you need to have the STATA software installed in your local computer and after starting the program you can follow the guidelines in chapter 4 (data analysis).

3 Data Collection

This section contains a detailed description of how the data for the sequential analysis is gathered and stored. This includes two subsections the (1) tweets data and the (2) bitcoin price data.

3.1 Tweets Data

Whit the python script real-time twitter data are streamed and stored. This happens with help of a raspberry pi.

3.1.1 Python Script

Twitter offers different Application Programming Interfaces (API) for collecting data. However, the time frame for gathering data on a freely base with full access is limited to 7 days (**twitterinc2017a**). On the other hand, python offers different twitter libraries. Such as the open-source package tweepy. This package has been used for streaming the twitter data as it simplify the script.

Installing Tweepy

Tweepy is installed very simply by running following commands in the command prompt.

```
pip install tweepy
```

If the previous downloaded python installation package does not contain the tweepy library, the tweepy package has to be downloaded. The package can be downloaded for free from the following link:

```
https://pypi.python.org/pypi/tweepy
```

Twitter Authentication

To access the twitter data, twitter requests an identification of the user. The identification is assured by different keys and access tokens. Those are the (1) consumer key (CKEY), (2) consumer secret (CSECRET), (3) access token (ATOKEN) and (4) access token secret (ASECRET).(**twitterinc2017b**)

To obtain the mentioned keys and tokens a twitter account is needed. Once, a twitter account exist a application has to be created. This has to be conducted by login with the twitter account credentials under the following link:

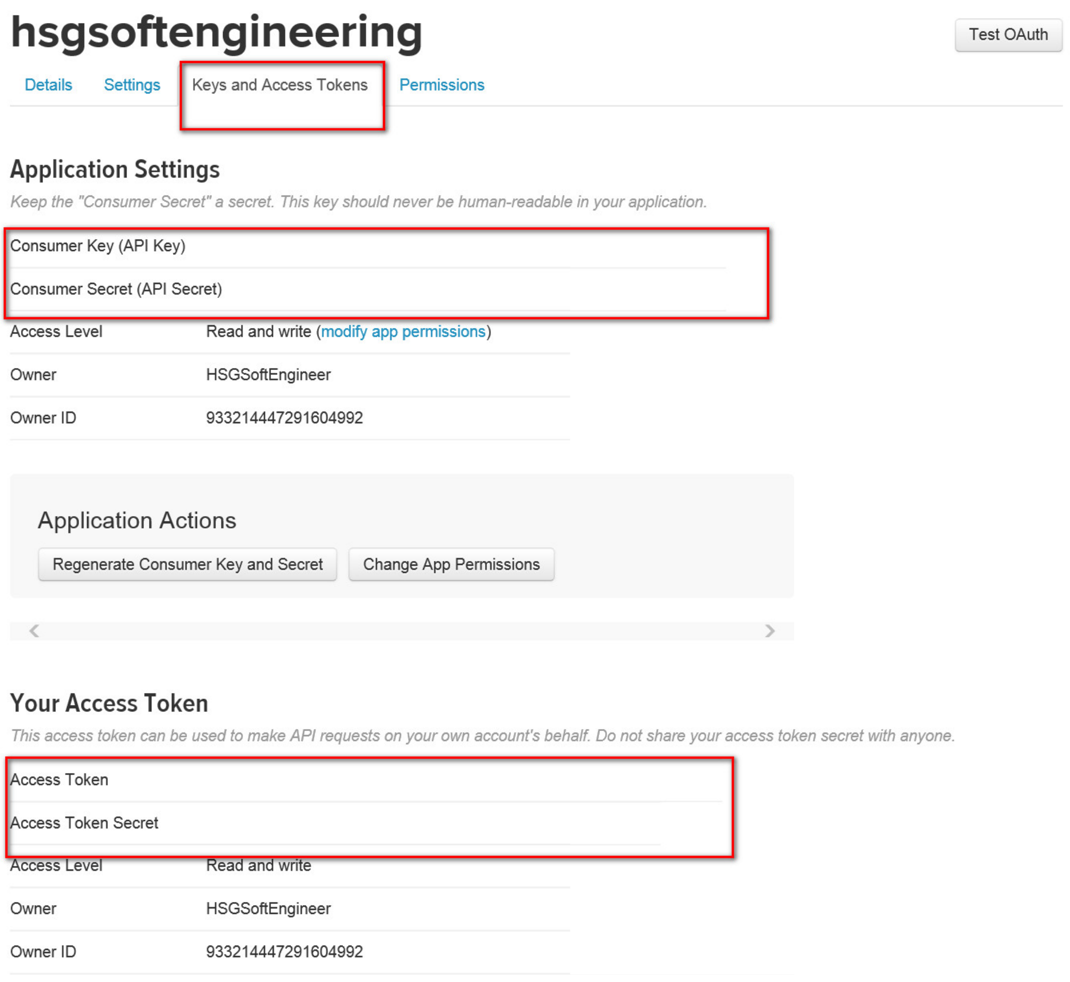
```
%https://apps.twitter.com/
```

After the creation of a application, the keys and tokens can be extracted. Figure 1 illustrates how to retrieve the keys and tokens.

Twitter Streaming API

By running the python script `collectTwitterData.py` real-time twitter data is pushed in a Java Script Object Notation (JSON) format. Tweets are just pushed in case a tweet contains the defined key word `bitcoin`.

```
$ python collectTwitterData.py
```


hsgsoftengineering

Test OAuth

Details | Settings | Keys and Access Tokens | Permissions

Application Settings

Keep the "Consumer Secret" a secret. This key should never be human-readable in your application.

Consumer Key (API Key)	
Consumer Secret (API Secret)	
Access Level	Read and write (modify app permissions)
Owner	HSGSoftEngineer
Owner ID	933214447291604992

Application Actions

Regenerate Consumer Key and Secret | Change App Permissions

Your Access Token

This access token can be used to make API requests on your own account's behalf. Do not share your access token secret with anyone.

Access Token	
Access Token Secret	
Access Level	Read and write
Owner	HSGSoftEngineer
Owner ID	933214447291604992

Figure 1: Twitter Keys and Access Tokens, based on **twitterinc2017c**.

From the JSON format the following parameters are decoded:

- created at: Timestamp of the created tweet
- text: Text of the tweet

The timestamps are in UTC time.

After a successfully decoding the parameters the data is appended to the `twitterData.csv` file in a new row and saved in the folder `/data`.

An excerpt of an example response looks like the following:

```
[
    {
        "created_at": Tue Dec 19 20:40:48 +0000 2017,
        "text": "RT @WhiteBitcoin: New Bitcoin White. Effective, Flexible,
            Reliable",
    },
    {
        "created_at": Tue Dec 19 20:40:49 +0000 2017 ,
        "text": "As #bitcoin prices surge, #xrp remains affordable and FAST!",
    },
    {
        "created_at": Tue Dec 19 20:40:49 +0000 2017,
        "text": "Singapore issues bitcoin warning after price rise - #BTC #Bitcoin
           #Crypto",
    }
]
```

Furthermore to ensure a high quality data set an exception has been integrated into the `collectTwitterData.py` script. If an error occurs an email will be sent with the error message. Furthermore the logfile will be appended to `/data/tweet_collection_error_log.csv`.

3.1.2 Hardware Setup

As explained in the previous section the twitter API was used to filter tweets in real time based on the buzzword ”bitcoin”. The Python program was running for seven days without interruption on a Raspberry Pi 2 (RPI) which was accessed with a Secure Shell (SSH) connection. This approach allowed us to ensure a continuous data stream while being able to access the program output regardless of time and location. Furthermore, due to a low energy consumption of the RPI the method is also the most cost efficient one compared to a personal computer or NAS Server. The following section describes the different steps that have been taken in order to setup the RPI 2, SSH, Git and Python3.

1. Setup of the Raspberry Pi
   (a) Burn image 2017-11-29-raspbian-stretch.zip to micro SD card available from
   ```
   https://www.raspberrypi.org/downloads/raspbian/
   ```
   (b) Boot image from micro SD card
   (c) Open SSH session using PuTTY SSH client for Windows available from
   ```
   https://www.chiark.greenend.org.uk/~sgtatham/putty/latest.html
   ```
   (d) Login to server with credentials
2. Update system to the newest software
   (a) Update list of applications with the command
   ```
   sudo apt-get update
   ```

   (b) Update applications with the command

```
sudo apt-get upgrade
```

3. Setup Git

   (a) Install Git using the command

```
sudo apt-get install git
```

   (b) Clone Git repository from

```
https://github.com/joelsonderegger/twitterbitcoin.git
```

4. Setup Python3

   (a) Install Python3 using the command: sudo apt-get install Python3

   (b) Install the tweepy module for python using the command

```
pip install tweepy
```

5. Collect Twitter Data

   (a) Open a new screen using the command

```
screen -S 'name'
```

   (b) Start Python program using the command

```
python3 collectTwitterData.py
```

   (c) Detach the screen to keep the program running after secure shell connection has been terminated using the commands

```
Ctrl+A
Ctrl+D
```

   (d) Re-attach the screen to check whether the script is still running using the command

```
screen -r 'name'
```

6. Download the collected data

   (a) Install FileZilla available from

```
https://filezilla-project.org/
```

(b) Connect to rpi and navigate to data location

(c) Download data manually

In summary, this approach worked well and ran smoothly. A clean setup is recommended to ensure a continuous data stream over a longer period.

3.2 Bitcoin Price Data

We wrote a Python script which collects Bitcoin price data as there was no preexisting data set that satisfied our needs. The Bitcoin price is best expressed by the Bitcoin Price Index. The Bitcoin price index (BPI) is an index of the exchange rate between the Bitcoin (BTC) and the US dollar (USD) (**kristoufek2015main**). The objective of the script was to gather hourly Bitcoin Price Index data for at least the time period in which we gather the tweets data. We found the an API by bitcoinaverage.com which sufficed our needs. An API description follows later.

3.2.1 Execution

By executing the python script `collectCryptocurrencyData.py` hourly data for the Bitcoin Price Index is retrieved.

```
$ python collectCryptocurrencyData.py
```

3.2.2 Output

After successfully running the python script `CollectCryptocurrencyData.py` the file `bpi.csv` is generated in the folder `/data`. It is important to note that every execution of the script overwrites any existing `bpi.csv` file.

The file `bpi.csv` contains historical Bitcoin Price Index data for one month on an hourly basis. Each data point consists of the following parameters:

- time: Timestamp on an hourly basis in UTC time
- average: Average price (in USD)
- high: Highest price (in USD)

- low: Lowest price (in USD)
- open: Opening price (in USD)

3.2.3 API: Bitcoinaverage.com

Bitcoinaverage.com offers a free API that provides real-time and historical price data for a range of crypto-currencies including Bitcoin. The following requests delivers data for an per hour monthly sliding window.

Request
The request to get the data for an per hour monthly sliding window looks as follows. This request require authentication that requires registration and the generation of an API key. The registration and generation of an API key is freely available on bitcoinaverage.com. The collectCryptocurrencyData.py already contains the necessary keys. This means that you need no register or generate keys to execute the script collectCryptocurrencyData.py.

```
https://apiv2.bitcoinaverage.com/indices/global/history/BTCUSD?period=monthly&?format=json
```

Response An excerpt of an example response looks like the following:

```
[
   {
      "high": 8271.04,
      "average": 8247.83,
      "open": 8242.39,
      "low": 8217.72,
      "time": "2017-11-22 15:00:00"
   },
   {
      "high": 8246.82,
      "average": 8203.19,
      "open": 8203.81,
      "low": 8157.25,
      "time": "2017-11-22 14:00:00"
   },
   {
      "high": 8267.27,
      "average": 8238.62,
      "open": 8248.77,
      "low": 8198.54,
      "time": "2017-11-22 13:00:00"
```

```
    }
]
```

4 Data Aggregation and Data Wrangling

After the tweets and BPI data set is generated needs to be aggregated before it can be analyzed. In addition, some data wrangling is necessary to bring the data in a format which than can be analyzed. The script `aggregateTwitterBpi.py` does all this.

4.1 Process

The process of the script `aggregateTwitterBpi.py` works as follows:

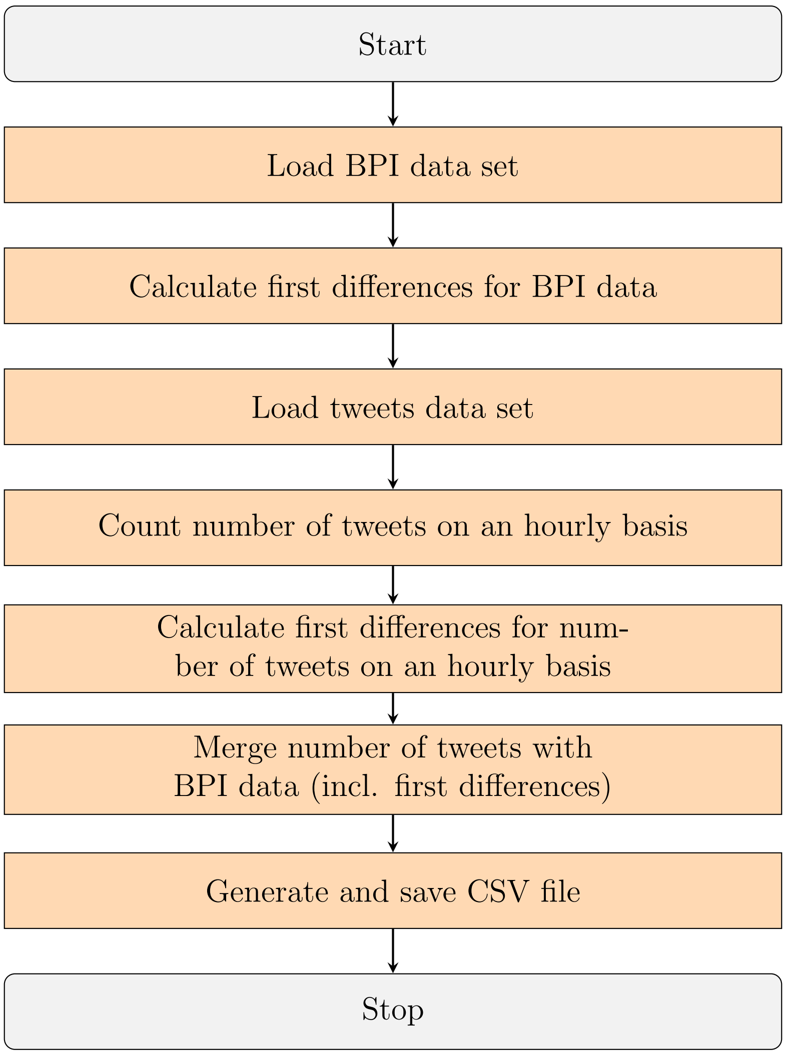

Start
Load BPI data set
Calculate first differences for BPI data
Load tweets data set
Count number of tweets on an hourly basis
Calculate first differences for number of tweets on an hourly basis
Merge number of tweets with BPI data (incl. first differences)
Generate and save CSV file
Stop

Figure 2: Data aggregation and data wrangling process

Here a short description of what happens at the various steps:

Load BPI data set The script takes the BPI data set that is located at `data/bpi.csv`.

Calculate first differences for BPI data The first differences of the BPI closing prices and the natural logarithm of the first differences of the BPI closing prices are calculated. The reason for calculating these values is given in the data analysis section.

Load tweets data set The script takes the tweets data set that is located at `data/twitterData.csv`.

Count number of tweets on an hourly basis First, groups the tweets by hour. Second, counts the number of tweets for every hour.

Calculate first differences for number of tweets on an hourly basis The first differences of the number of tweets and the natural logarithm of the first differences of the number of tweets are calculated. Again, the reason for calculating these values is given in the data analysis section.

Merge number of tweets with BPI data (incl. first differences) Insert Image

Generate and save CSV file Finally, a CSV file is generated with the merged data.

4.2 Execution

By executing the Python script `aggregateTwitterBpi.py` a aggregated data set that contains number of tweets and Bitcoin Price Index, both on an hourly basis, gets generated.

```
$ python aggregateTwitterBpi.py
```

4.3 Output

After successfully running the Python script `aggregateTwitterBpi.py` the file `nr_of_tweets_bpi_closin` is generated in the folder `/data`. It is important to note that every execution of the script overwrites any existing `nr_of_tweets_bpi_closing_price.csv` file.

The file `aggregateTwitterBpi.csv` contains historical tweets and Bitcoin Price Index data. Each data point consists of the following parameters:

- time: Timestamp on an hourly basis in UTC time
- nr_of_tweets: Sum of tweets
- df_nr_of_tweets: First difference of the sum of tweets
- log_df_nr_of_tweets: Natural logarithm of the first difference of the sum of tweets

- bpi_closing_price: BPI closing price
- df_bpi_closing_price: First difference of the BPI closing price
- log_df_bpi_closing_price: Natural logarithm of the first difference of the BPI closing price

5 Data Analysis

The data analysis was performed with the statistical software STATA. The corresponding stata file and .do file are stored in the folder statistical_analysis_stata. Following we describe all the commands used to produce our econometric model.

Step 1:
After starting data the first step was to import the data file (.csv file) into stata. We did so by following command (be carefull your location of the .csv file may differ):

```
 import delimited "your_location.csv"
```

Step 2:
STATA is not recognizing our variable "time" as a date variable and therefore we create a new variable "time2" and giving it the same values as "time" and giving it the same format as "time" has. At the end we define "time2" to be a time series with an hourly progress

```
 gen double time2 = clock(time, "YMD hms")\\
format time2 %tcNN-DD-CCYY_HH:MM:SS\\
order time2, after(time)\\
tsset time2, format(%tcNN-DD-CCYY_HH:MM:SS) delta(1 hours)
```

Step 3:
Checking for stationarity for all variables by using the Augmented Dickey-Fuller (ADF) test with a time delay of 0 periods (lag 0).

```
dfuller nr_of_tweets, lags(0)
dfuller df_nr_of_tweets, lags(0)
dfuller log_df_nr_of_tweets, lags(0)
dfuller bpi_closing_price, lags(0)
dfuller df_bpi_closing_price, lags(0)
dfuller log_df_bpi_closing_price, lags(0)
```

Step 4:

Calling the lag-order selection criteria statistics for our VAR model. The maximum amount of lags is 7 and the variables that we chose for the latter VAR model are df_nr_of_tweets and df_btc_closing_price.

```
varsoc df_nr_of_tweets df_btc_closing_price, maxlag(7)
```

Step 5:
Running the VAR regression for our two variables and recommended lag of 2 (from Results of Step 4).

```
var df_nr_of_tweets df_btc_closing_price, lags(1/2)
```

Step 6:
Calling the granger causality test with lag length 2.

```
Vargranger
```

Additionally:
The Regression Results were exportet to a latex format by instaling and using the outreg2 package. This piece of latex code was then inserted into in the paper. All other results we edited with an image software and inserted them as .png file into the paper

```
ssc install outreg2
outreg2 using var_gegression_results.tex, replace
```