University of Zurich[UZH]

Mining concentration and capital accumulation in Bitcoin

Bachelor's Thesis

Submitted in partial fulfillment of the requirements for the degree of Bachelor of Banking and Finance

Author

[Dario Bugmann]

[Fuchsmättliweg 6]

[5106 Veltheim]

[15-708-852]

[dario.bugmann@uzh.ch]

Supervisor

Prof. Dr. Claudio J. Tessone

Department of Network Science

University of Zurich

Date of Submission: [17 January 2019]

Executive Summary

Invented in 2008 with the release of the white paper by a person under the pseudonym of Satoshi Nakamoto, Bitcoin is the most famous cryptocurrency. One key difference between Bitcoin and a regular currency is that Bitcoin is decentralised. This means that no government or other centralised institution is behind or in charge of this currency. All transactions are verified through the network, where every transaction is registered in a public ledger. Bitcoin is called a cryptocurrency because it uses public key cryptography to make transactions. Transactions are made through secure digital wallets containing one or more addresses. This personal wallet mathematically ensures that money sent from or received into this wallet actually goes to the right person. These transactions are then bundled and stored as blocks in a public ledger called the blockchain. A new block is added through the process of mining, which requires a lot of computational power. As a result, different users pool their power and form a mining pool. The publicly stored data of the mining pool information for each block is used to calculate the monthly mining concentration, resulting in the three largest mining pools accumulating approximately 50% of the entire network's mining power. By defining four different cases, miners' addresses are identified and combined into users. Very high levels of inequality are found using the wealthiest miners' shares, as well as the Gini coefficient with values exceeding 0.8. By combining mining concentration and wealth inequality, a very strong, positive correlation is found when not controlling for confounding variables. It appears that time impacts either of the two variables, and thus they do not link in a causal way.

Contents

List of Figures

List of Tables

Listings

Chapter 1

Introduction to Bitcoin Technology

Cryptocurrencies, and Bitcoin in particular, have been a hot topic in both academic circles and public media since Satoshi Nakamoto first released a paper titled 'Bitcoin: A Peer-to-Peer Electronic Cash System' in 2008. Despite the increased popularity of Bitcoin, there are still many topics that have not yet been researched in depth. This is not surprising since the idea of Bitcoin has only existed for roughly 10 years.

The original idea of Bitcoin was to create a decentralised payment system (Nakamoto 2008). Since this is one of the major benefits of this cryptocurrency, it is important to analyse how the decentralisation developed. In 2013, the six largest mining pools had 75% of the total computing power of the Bitcoin network (Gervais et al. 2014). Being in control of over 50% of the Bitcoin network makes it possible to manipulate the proof-of-work and use it for malicious attacks such as double-spending attacks (Kroll, Davey, and Felten 2013). By manipulating the proof-of-work and holding the majority of the network at the same time, mining pools would be able to change transactions and even validate them by finding later blocks. Eyal and Sirer (2018) have demonstrated that the current threshold is close to zero when not applying their proposed solution. This means that even if the network power of a selfish mining pool is close to 0%, it is always more profitable for miners to join the selfish pool rather than a honest one. This will then most likely result in this selfish mining pool growing and eventually controlling the entire network. Furthermore, Eyal and Sirer have demonstrated that even with their solution applied, the Bitcoin protocol will never be safe if there is a selfish mining pool controlling over 33% of the network power. This is why the mining concentration of Bitcoin is an important and interesting topic to research.

The other aspect analysed in this thesis is the wealth accumulation of Bitcoin. Kondor et al. (2014) have done some research in this field; however, their method was address-based and applied to the entire Bitcoin system. The goal is to analyse the wealth accumulation based on users, not addresses. Also, this thesis focuses on the wealth aggregated simply by the miners, a subset of the entire user base. This makes it necessary to set parameters and develop different methods to identify

the miners. The effects of large-scale inequality in an economy are drastic. Wilkinson (2011) has chosen inequality as a measure over poverty and found correlations between income inequality and mental illnesses, drug use, obesity, and life expectancy. Kelly (2000) has come to the conclusion that higher economic inequality results in higher crime rates, even when controlling for poverty, race, and family composition. Inequality was also found to negatively correlate to happiness because of higher perceived unfairness and less trust in other people in years with more inequality (Oishi, Kesebir, and Diener 2011). For these reasons, examining miner inequality in the Bitcoin ecosystem is important, especially considering the increase of cryptocurrencies' popularity.

After addressing both of the aforementioned topics, determining relations between them is the third and final part of this thesis. The relation between the inequality of the miners and the mining concentration is explored based on the assumption of discovering a positive correlation between mining concentration – and therefore centralisation – and inequality between miners.

'What are the patterns of mining concentration and capital accumulation of Bitcoin?' combines all of the aforementioned topics and is the research question for this thesis.

The thesis is divided into four sections. Chapter 1 presents the basic concepts of Bitcoin, blockchain, and mining pools, which build the technical basis and allow for further examination of the research question. Chapter 2 combines both the data and the methodology applied to answer my research question. Chapter 3 presents the results in written form with the help of some illustrations. The results are also discussed in this chapter. To conclude this thesis, Chapter 4 presents the answers to the research question and reviews the research conducted.

If not mentioned separately, the rest of this technical Chapter 1 mainly relies on Franco (2015).

1.1 Cryptography

Bitcoin is a cryptocurrency. The name already suggests how relevant cryptography is to Bitcoin. The two main cryptographical instruments used in relation to Bitcoin are hashing and digital signatures. The latter requires an explanation of standard public key cryptography.

1.1.1 Public Key Cryptography

Public key cryptography is asymmetric, meaning that there are two different keys: a public key and a private key. The main principle is that while the public key is open to other users, the private key is always kept private. There are multiple algorithms available to create a key pair, but what is most important is that the public and private key are mathematically linked.

If Alice wants to send a message to Bob, Bob must first generate a key pair using a key generation algorithm. He then sends his public key to Alice over an insecure channel such as the internet, and Alice encrypts her message using Bob's *public key*. She then once again uses an insecure channel to send the message back to Bob. Bob is now able to decrypt Alice's encrypted message with his *private*

key. Because of the insecure channel between Alice and Bob, a potential security problem called a *man-in-the-middle attack* could occur. Imagine that there is a third person called Claire who is able to intercept Bob's public key. Claire then also generates her own key pair and gives her public key to Alice, who thinks that it is Bob's public key. Alice then encrypts her message using Claire's public key and sends it back to Bob. Claire once again intercepts the encrypted message, which she decrypts using her private key. She then modifies the decrypted message as she desires and encrypts it using Bob's public key. Bob receives the message and decrypts it using his private key. The problem with this attack is that neither Alice nor Bob knows that there has been a man in the middle modifying the message.

There are multiple ways to try to avoid this problem, such as a *web of trust* or a *public key infrastructure*, but with Bitcoin in mind, this thesis examines digital signatures.

1.1.2 Digital Signature

The issue with public key cryptography so far is that there is no certainty about the identity of the sender (anyone could have used the public key provided). This is where digital signatures come into play. The main idea of a digital signature is the same as that of a regular, handwritten signature. It is used to ensure that the message sent is coming from the person who signed it and has not been edited.

If Alice wants to sign a message, she first sends her public key to Bob,[1] then signs the message with her private key and sends the signed message to Bob. If Bob is able to verify Alice's signature using her public key, he can be sure that the message was originally from Alice and was not tampered with. Thus, a digitally signed message has the characteristics listed below.

Authentication: Because the sender is the only person in charge of the private key, the receiver can be certain that the message comes from the expected sender.

Non-repudiation: The sender cannot at any later point in time deny having signed the message, because it is assumed that the sender's private key is kept private at all times.

Integrity: The signed message could not have been modified, because any change would result in the signature being invalid.

As far as Bitcoin is concerned, it uses the elliptic curve (EC) for its public key algorithm. The digital signature algorithm (DSA) is used for the digital signature. Combined, these two algorithms are also called the elliptic curve digital signature algorithm (ECDSA).[2]

[1] It should be noted that a potential man-in-the-middle attack could also take place here. This problem does not exist with Bitcoin, because the hashes of public keys are stored in the blockchain.

[2] For more information, consider Johnson, Menezes, and Vanstone (2001).

1.1.3 Cryptographic Hashing

Hashing is a very relevant subject in relation to cryptocurrencies, and Bitcoin mainly uses a hashing function called Secure Hash Algorithm 256 (SHA-256). The idea and possible problems of hashing are covered in this section.

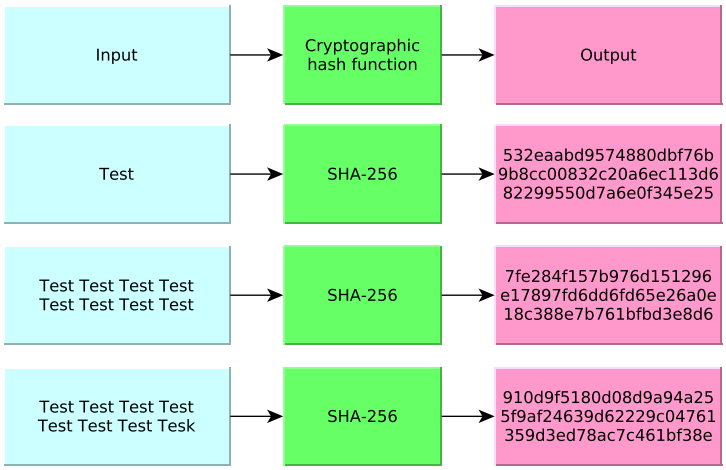


Figure 1.1 – Example of cryptographic hash function SHA-256

As shown in Figure 1.1, a hash function is a function that takes an input of variable length and creates an output of a set length. The output is usually called a *hash* or *hashvalue*. For example, SHA-256 always creates a 256-bit output. The advantage of such a function is obvious: no matter how long the input is, the size of the hash is always the same. This is extremely useful when trying to save space, but if one allows an input of arbitrary length and the output is limited in length, there have to be some inputs which, when hashed, return the same output. This is called a *collision*. If the hash function used is collision-resistant, this manipulation is very unlikely to happen, but not impossible (in fact, Stevens (2012) found a collision with the Message-Digest Algorithm 5 (MD5) within three weeks). Hash functions can also be used to encrypt messages because they are one-way functions, meaning that the original message cannot be reconstructed from the hash.

According to wikipedia.org (2018), a good cryptographic hash function has the following five properties:

1. It is deterministic, meaning that input x will always result in the same output y.
2. Calculating the hash is quick, independent of the message.
3. It is infeasible to calculate the message from the hash (i.e., it is a one-way function).
4. It is infeasible to find two different messages with the same hash value (collision).
5. A small change in the message should lead to a completely different hash value (see Figure 1.1).

1.1.4 Bitcoin Address

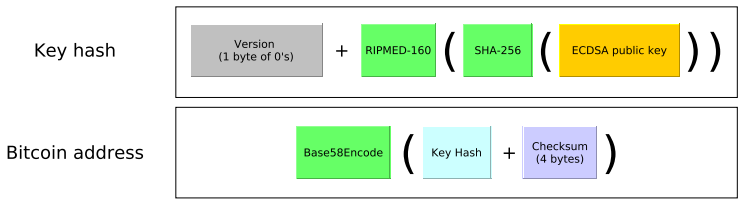


Figure 1.2 – Creation of a typical Bitcoin address, based on bitcoin.it (2018e)

A Bitcoin address is a hash of the ECDSA public key. To be exact, as shown in Figure 1.2, the public key is hashed using SHA-256 and then RACE integrity primitives evaluation message digest (RIPMED-160). The second hashing process breaks the hashed public key from 32bytes down to 20bytes. This second hashing involves a trade-off between making the transaction size smaller and still keeping collisions very unlikely. The resulting hash is then tied together with the version number, resulting in the key hash. The checksum (the first four bytes of the double-SHA-256 hash of the key hash) is then added to the key hash and hashed with Base58Encode,[3] resulting in the *Bitcoin address*. Not considering testnet addresses, there are two main address types:

Public key address: Has a version value of zero (decimal), which leads to an address beginning with one once encoded.

Script address: Has a version value of five (decimal), which leads to an address beginning with three once encoded.

The different types of addresses become of great importance when examining the different verification mechanisms. This is explored in Section 1.3.3. According to btc.com (2018b), public key addresses are still the most common ones, but script addresses have also gained significant popularity. This thesis primarily focuses on public key addresses; nevertheless, an overview of script addresses is provided.

1.2 Network

The central Idea of Bitcoin was to create an electronic peer-to-peer (P2P) (see Figure 1.3) cash system without the need to trust other people or institutions (Nakamoto 2008). A P2P network differentiates itself from a classical, server-based network. It does not have one central entity storing data; instead, every peer (also called *node*) in the P2P network acts as a storage and verification unit and exchanges data with other nodes in the network. According to Nakamoto (2008), today's financial institutions cannot avoid disputes that lead to the reversal of some transactions. As a consequence, the need for trust in these institutions becomes greater, but without a trustworthy third party, it is very expensive

[3] This hashing algorithm deletes some signs that are difficult for the human eye to distinguish.

to search for additional information about one's transaction partner in order to avoid fraud. Using physical cash can prevent these costs and the uncertainty of a payment potentially being reversed. Over a communication channel, however, this is not guaranteed. This is where Bitcoin comes into play. The goal of Bitcoin was to create an electronic payment system based on cryptographic proof, not just on trust, that allows two parties to trade without requiring a trusted third party (Nakamoto 2008).

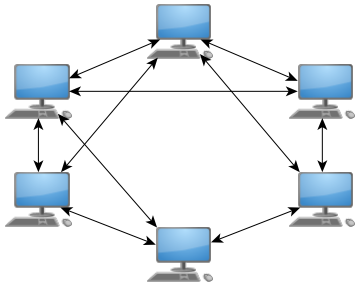


Figure 1.3 – A peer-to-peer (P2P) network

Before going into more detail about some specific parts of Bitcoin technology, it is essential to see the larger picture of how the different aspects come together. The following procedure is based on Nakamoto (2008):

1. New transactions are broadcast to all nodes.
2. Each node collects new transactions into a block.
3. Each mining node works on finding a difficult proof-of-work for its block (see Section 1.5.1).
4. When a mining node finds a proof-of-work, it broadcasts the block to all nodes.
5. Nodes accept the block only if all transactions in it are valid and not already spent and the proof-of-work has been applied.
6. Mining nodes express their acceptance of the block by working on creating the next block in the chain, using the hash of the accepted block as the previous hash.

Note: A *node* is any computer connected to the network. A *mining node* is a subnode that behaves the same as a regular node but also mines.

There are two different types of nodes in the network: full nodes and lightweight nodes. Full nodes store the entire blockchain and verify transactions as listed above, whereas lightweight nodes do not store blocks and only check if a certain transaction is included in the block with the help of simplified payment verification (SPV) (see Section 1.4.1).

1.3 Transaction

A transaction is used to change the owner of a bitcoin. The transaction itself is not encrypted,[4] meaning that every node can check and see the transaction once it is broadcast to the network.

```
Input:
Previous tx: f5d8ee39a430901c91a5917b9f2dc19d6d1a0e9cea205b009ca73dd04470b9a6
Index: 0
scriptSig: 304502206e21798a42fae0e854281abd38bacd1aeed3ee3738d9e1446618c4571d10
90db022100e2ac980643b0b82c0e88ffdfec6b64e3e6ba35e7ba5fdd7d5d6cc8d25c6b241501

Output:
Value: 5000000000
scriptPubKey: OP_DUP OP_HASH160 404371705fa9bd789a2fcd52d2c580b65d35549d
OP_EQUALVERIFY OP_CHECKSIG
```

Listing 1.1 – Example of a basic Bitcoin transaction (bitcoin.it 2018g)

An input of a Bitcoin transaction is a reference to an unspent output previously received by a user (Previous tx in Listing 1.1). The index provided indicates at what position the unspent output is located in the previous transaction. In this example, it is the first output of the previous transaction. Also, there is a variable scriptSig in the inputs of a transaction. It contains two parts, the first being the signature of the sender and the second being a public key corresponding to the signature.

The first field in the output is the value (in satoshi)[5] that is about to be spent. In the output there is also a variable containing a script, namely scriptPubKey. In addition to some commands, which are examined later in the thesis, the destination address for the coins is part of the script.

Because an input of a transaction is an output of a previous transaction, there is often change at the end of the new output going back to the sender of the transaction. For example, If User 1 wants to send two bitcoins (BTC) to User 2, but he previously received three BTC in another transaction, he then has two new outputs: two BTC to User 2, and one BTC to himself.

1.3.1 Fees

Whenever all combined inputs of a transaction have a greater value than all the outputs, the difference is by definition a transaction fee. All the transaction fees add up and become an additional mining reward to whoever manages to mine the corresponding block.

It is important to have transaction fees because they serve as incentives for the miners. Without them, there would be no interest for a miner to include the transaction in the block they are working on, and thus the transaction would never be included in any block.

[4] Nevertheless, it contains a signature as shown in Listing 1.1 (scriptSig).
[5] 1 satoshi = 0.00000001 BTC

1.3.2 Signing a transaction

The original transaction is copied and shortened before the copied transaction is double-SHA-256 hashed. Depending on the type of the signature, only the parts that have to be signed are left on the copy after the shortening. This hash is then signed using the sender's private key.

Every signature in ScriptSig includes a flag called a hash type. The hash type indicates which inputs and outputs have been signed. Based on bitcoin.org (2018b), a list of the possible types is provided in Appendix B.

1.3.3 Verification

After adding the signature, the transaction is sent to a network node. If it is verified, the node sends it to other nodes in the network. The following elements are then checked:[6]

1. Inputs have not yet been spent (a special database is used for this purpose).
2. $\sum Inputs \geq \sum Outputs$
3. The validity of the signature. This is examined in more detail below.

The most standard Bitcoin transaction uses public key addresses (addresses beginning with one) and therefore uses the *Pay-to-PubKeyHash* method. Listing 1.2 shows the verification script for those addresses.

```
scriptPubKey: OP_DUP OP_HASH160 <pubKeyHash> OP_EQUALVERIFY OP_CHECKSIG
sciptSig: <sig> <pubKey>
```

Listing 1.2 – Pay-to-PubKeyHash verification algorithm (bitcoin.it 2018g)

In order for a user to be able to create a new transaction, he/she must add unspent transaction outputs to the input of the new transaction and prove that he/she owns the corresponding private key/s. This verification process is demonstrated in detail in Appendix A. If this script returns true, the transaction is valid and is therefore sent to the network.

If one has privacy and security in mind, it is recommended to use a different address for every transaction made because with Bitcoin the address of the sender is not made public until he/she actually broadcasts the transaction to the network. The more often a certain address is used to make a payment, the easier it becomes to determine the private key that corresponds to the public key. Since the transaction is published and contains its output address as well as the signature (which was created with the private key), really powerful hardware, such as quantum computers, are capable

[6] This list is simplified and by no means complete. For more information, see `https://en.bitcoin.it/wiki/Protocol_rules#.22tx.22_messages`

of determining the private key. This is very difficult to do, but it becomes easier the more often a particular address is used, because to spend those outputs, one must provide the same private key over and over again.

Without going into too much detail, it is important to mention the *Pay-to-Script-Hash* Bitcoin transaction type, which was introduced with BIP 16.[7] Its validation works differently.

Unlike with the Pay-to-PubKeyHash transaction, the transaction is sent to a script hash address (beginning with three). The foundation of this verification process is a *redeem script*, which is held by the receiver of a transaction. It therefore allows the receiver to define the conditions to be met in order to spend the outputs again. The hash of this script results in the address, which is given to the sender. To be able to spend this transaction, the sender must include the script and some additional data (mostly signature/s) in his/her input (scriptSig). If that script hashes to the output of the previous transaction's output address and the additional data makes the script return true, the transaction is valid. The Pay-to-Script-Hash transaction type is mostly used for multi-signature transactions, which increase the level of security because they also allow for more complex verification mechanisms and are therefore most likely the future of blockchain technology.

1.3.4 Genesis Transaction

The very first transaction of every block is a special one: a so-called genesis transaction. It has only one input, which is not linked to any other previously received transaction and serves no purpose. It can be anything. Some miners use it to send a certain message. The output, however, has the same values as every other transaction and consists of the block reward (plus any additional transaction fees) and the address of the miner/s.

1.3.5 Wallets

Because sending a transaction manually every time one wants to send some bitcoins would be a very time-consuming and complex task, wallets are available. A wallet is software that does everything one would expect from a regular wallet with some differences in the way it works. Unlike a regular wallet, it does not actually store the money inside the wallet, because all the transactions are stored in the blockchain. What a wallet can do, however, is create many key pairs (and, as a result, addresses) that it safely stores on the owner's computer. The wallet then constantly checks the network for transactions that correspond to the addresses stored in the wallet. The result is that one has an available balance of bitcoins to spend. The wallet also helps to spend the received bitcoins: it creates the transaction, signs it, and sends it to the network for validation. Some wallets even create an address for every change transaction going out of the wallet. The safety of the wallet itself is the most important aspect and is achieved by encrypting the private keys.

[7] `https://github.com/bitcoin/bips/blob/master/bip-0016.mediawiki`

One could say that a wallet makes it possible for any user to fund or receive transactions without knowing the technical background of the Bitcoin network.

1.4 Block

Having examined a basic Bitcoin transaction, the next larger entity in the Bitcoin environment is called a block. According to bitcoin.it (2018a), a block has the following properties: magic number, block size, block header, transaction counter, and transactions. For the purposes of this thesis, the most important properties are that a block contains transactions and some additional information in the block header.

1.4.1 Block Header

The block header is of particular importance because it is used to deliver the proof-of-work. This process is explored in more detail in the mining section of this thesis. According to bitcoin.it (2018b), the block header has the following properties:

Version: The block's version number.

hashPrevBlock: The 256-bit hash of the previous block header.

hashMerkleRoot: The 256-bit hash based on all of the transactions in the block (indirectly hashed based on the merkle root).

Time: The current timestamp in seconds since 1970-01-01T00:00 UTC.

Bits: The current target number in a compact format (the hash of the block header has to be less than the target when mining); the lower the target, the harder the puzzle.

Nonce: The 32-bit number beginning at zero that was found by the successful miner.

The overarching concept behind the merkle root is that even though the transactions are not stored as is in the block header, they are stored indirectly. Following bitcoin.it (2018f), the merkle root hash is constructed as follows: using the transaction identifications (txids) created by individually hashing all the transactions in the block, the merkle tree is constructed by pairing each txid with one other txid before hashing them together. This procedure collapses two txids into one. If there is an odd number of txids, the txid left over is hashed with a copy of itself. In Figure 1.4, this means that Tx0 and Tx1 are hashed to obtain their txids. Those txids are then hashed together again until only one hash remains. This is called the *merkle root*.

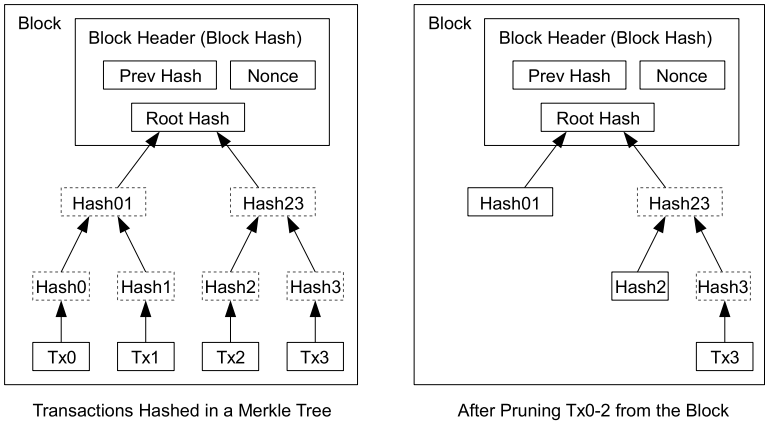


Figure 1.4 – Construction of the merkle root for the block header (Nakamoto 2008)

The concept of a simplified payment verification (SPV) uses the merkle root and merkle tree in order to verify transactions without storing the entire blockchain on the local computer, which is essentially what a full peer would do. Keeping Figure 1.4 in mind, if one wishes to verify that Tx3 was included in the block, one would only need to determine the hash of Tx2 (Hash2), Hash01, and the merkle root hash from a full peer. Any full peer can be used for gathering the block header and neighbouring hashes because manipulating the block header is very expensive and faking the neighbouring txids would result in a failure of the verification process (bitcoin.org 2018c).

1.5 Blockchain

A problem that occurs within the system as explained so far is that a recipient can only verify the change of ownership. One is able to follow the flow between the sender and recipient of a transaction, but one cannot to be guaranteed that a particular recipient was the first one to receive this exact coin. For instance, the output of a transaction could have already been spent before. In order to prevent double-spending, a system is needed where transactions are publicly announced and all the users of this system agree upon a single history of transactions (Nakamoto 2008). This history is called the blockchain.

Transactions are bundled together to Block 1 and hashed. A history of transactions is created by linking this hash to the previous block. When creating Block 2, the hash of the previous block, Block 1, is added to the data of Block 2. Block 2 is now hashed and published. An attacker wanting to change a transaction from Block 1 is now also required to redo the proof-of-work for all the following blocks in the blockchain, which becomes difficult with every block added to the blockchain. Chaining the blocks in Bitcoin is achieved by including the hash of the previous block header, which includes the merkle root and thus all transactions of that block, in the new block header (see Figure 1.5 and Section 1.4.1). The Bitcoin protocol defines that the correct blockchain is the longest, in terms of accumulated difficulty, available chain. It is possible that there exist multiple chains; for instance,

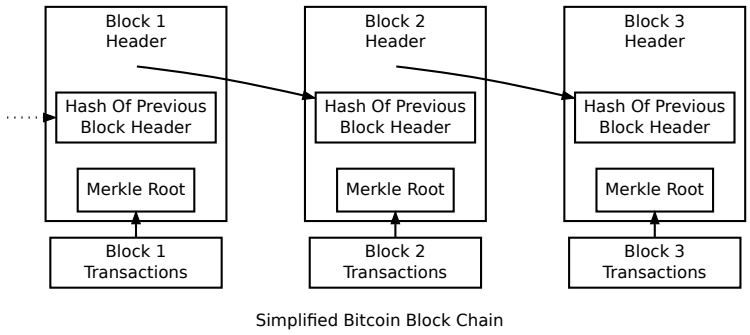


Figure 1.5 – Simplified blockchain overview (bitcoin.org 2018a)

when two new blocks are mined within a short period of time. In that case, not all mining nodes receive the same new block over the network, and, because the two chains are of same length, the miners do not know which chain is the correct one. Therefore do not work on the same chain for some time. As soon as one chain becomes longer again, most miners will switch to that chain, because work done on the other chain would be wasted.

1.5.1 Mining

Mining is the process of adding newly created blocks to the blockchain. In order to do this, miners have to create new blocks that contain a proof-of-work.

> 'Once the CPU effort has been expended to make it satisfy the proof-of-work, the block cannot be changed without redoing the work. As later blocks are chained after it, the work to change the block would include redoing all the blocks after it.' (Nakamoto 2008, p. 3)

A proof-of-work is a security measure required to prevent the blockchain from being overburdened with newly created blocks. In the timestamp example from above, this proof-of-work involves not just calculating the hash but solving a special condition in order to obtain a certain hash structure. The proof-of-work consists of finding a nonce (begins at zero and increases linearly) that, when hashing the block header (see Section 1.4.1), returns the right amount of zero bits at the beginning. The amount of zero bits to be matched is defined by the current target number and determines the difficulty: the more zero bits required, the more difficult it is to find a valid proof-of-work. This procedure is shown in Figure 1.6. The algorithm used by Bitcoin to complete this process is called *hashcash*.

It is crucial to mention here that the block header may vary from miner to miner because the genesis transaction differs. This results in a different hashMerkleRoot for the miners, and thus other nonces are required in order to solve the condition. Also, other transactions may be added to the block and change the hashMerkleRoot. In conclusion, because of different hashMerkleRoots, every nonce a user tries has the same chance of winning, meaning that the reward is not guaranteed to the user

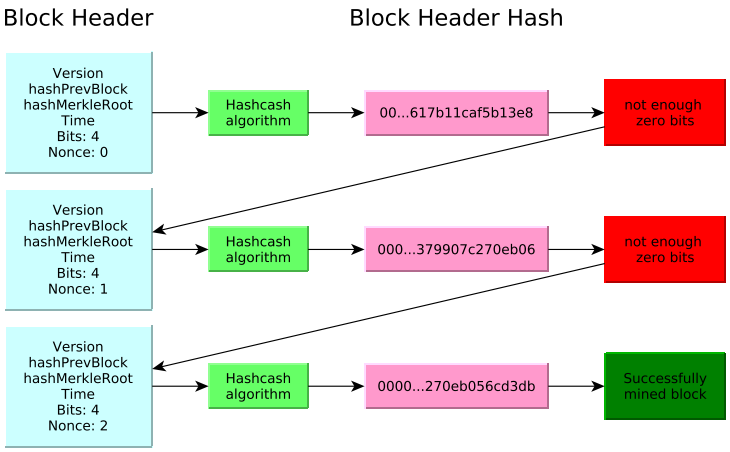


Figure 1.6 – Simplified demonstration of the Bitcoin proof-of-work

who can go through the nonces the fastest and thus always stay ahead of the others. The reward for mining a block begins at 50 BTC and halves every 210,000 blocks. In order to keep the halving of the reward at a constant pace, the difficulty of mining a block is adjusted regularly (every 2,016 blocks). The goal is to have on average six newly mined blocks added to a blockchain every hour, and thus the reward is expected to halve every four years. The essential aspect here is that finding the correct nonce is very difficult and requires significant computational power in order to be found, whereas verifying only requires one hash to be calculated. An example of a simplified mining process is shown in Figure 1.6.

The hardware for mining bitcoins has changed significantly over the years. When Bitcoin was first created, a central processing unit (CPU) provided enough computational power to mine. Until the summer of 2010, mining was done using CPUs exclusively. The best CPUs at that time had a hash rate of about 20MH/s. Beginning in mid-2010, graphics processing unit (GPU) mining became the new standard. Graphics processing units offer much higher hash rates (a high-end card Nvidia Geforce GTX 590 is able to pull just shy of 200 MH/s (bitcoin.it 2018d)). For a rather short time, field-programmable gate arrays (FPGAs) were competition for GPU mining. They had the advantage of being more power-efficient, but they were very expensive: in terms of cost per GH/s, they were behind the current GPUs (Taylor 2013). The latest generation of hardware used for Bitcoin mining is called application-specific integrated circuit (ASIC). Application-specific integrated circuits were specifically built to perform the hash calculations required for mining bitcoins more efficiently than any other prior technology. At the time of writing hashing, common hash rates are 10TH/s or more, which is 50,000 times the hash rate CPUs originally offered (bitcoin.it 2018c). The increasing difficulty of mining results in any other mining hardware no longer being competitive or profitable.

The computing power required for solving the proof-of-work makes it really expensive and difficult to change a transaction from the past. In addition, the probability of also redoing all the blocks coming

after the changed one and becoming the new longest chain[8] reduces exponentially with every new block added to the current longest chain (Nakamoto 2008). This makes the blockchain technology tamper-proof.

1.5.2 Mining Pools

With the increasing difficulty and accumulated hash rate of the entire network, it can take years for a solo miner to mine a block on his/her own (cryptocompare.com 2018). There is, however, a method that can address this problem.

> 'Consequently, miners typically organize themselves into mining pools. All members of a pool work together to mine each block, and share their revenues when one of them successfully mines a block. While joining a pool does not change a miner's expected revenue, it decreases the variance and makes the monthly revenues more predictable.' (Eyal and Sirer 2018, p. 97)

In conclusion, as a result of joining a mining pool, a user receives a much more predictable income. The income of a mining pool is shared proportionally[9] among its users. In order to start mining, most mining pool operators only deliver the block header to their miners, which is more efficient, because not every single miner has to download all the transactions that should be included in the block. Also, the pool owner is the only one deciding whether or not to include a transaction in the block. This could be problematic if the owner defines some unethical rules. The benefit of only giving the block header to the miners is that the recipient address of the genesis transaction belongs to the pool and cannot be changed by a miner. If this was not the case, an individual miner could have incentive to publish the block on his/her own, and therefore receive the whole reward, when he/she finds one. There are many different types of payout mechanisms used by mining pools, including pay-per-share (PPS), pay-per-last-N-shares (PPLNS), and pay-to-pool (P2pool). The basic concept of PPS is that every miner sends shares, which are valid proofs of work of a lower difficulty, to the mining pool. Every time the miner delivers a share to the mining pool, he/she is paid a flat rate. This shifts the risk of not receiving a block for a long time to the mining pool provider because he/she has to pay the miners whether or not a block is mined. Pay-per-last-N-shares shifts the risk of not finding a block for a long time back to its users. This usually leads to lower fees for the miners. Whenever a block is found, it pays out according to the last N shares of the pool. Assuming that $N = 100$ and a miner appears to have contributed 10 of those shares to the pool, when the pool mines a block, the miner receives 10% of the block reward. It should be noted here that it could happen that the miner stops mining and the next block is found very shortly after the last one. The miner could still have some shares in the last N shares and receive another reward for the same shares.

[8] The longest chain is the one with the most combined difficulty

[9] Considering the hash rate of each user.

On the other hand, the pool could also be unlucky and the miner could fail to receive a reward in spite of having distributed some shares. With paying out directly through the genesis transaction, the P2Pool approach is perhaps the most intuitive. A *share chain*, containing the rewards of all the individual miners, is created and payed out to the miners after a block has successfully been mined. For more efficiency, not the current but the users in the *previous* share chain are rewarded when a new block is mined. The finder of a new block publishes it on his/her own and the miners are rewarded directly through the genesis transaction. Because the share chain is distributed, and thus P2pool works without a pool operator, every miner is required to operate as a full node. Storing the entire blockchain locally is not very efficient and is the biggest disadvantage of P2pool mining.

With these traditional approaches, the pool keeps the fees included in the genesis transaction. Because transaction fees have risen lately, alternatives like full pay-per-share (FPPS) have come to market. In this case, expected transaction fees are added to the block reward (12.5 BTC at the time of this writing), increasing the total block reward by $\approx 1\%$ (btc.com 2018c).

A few pools mining the majority of blocks is usually rated as suboptimal by the community, because this gives the pool more power for potential attacks. It also makes it possible for the pool to decide which transactions should be included in the block they are currently trying to mine. This could lead to the pool only accepting transactions with very high transaction fees because they are part of the miners' revenue. Also, the miners of a mining pool often do not work as a full node, meaning that they do not validate blocks and transactions by themselves. For these reasons, this thesis begins by evaluating the concentration of Bitcoin mining pools over time.

Chapter 2

Data and Methodology

2.1 Data

2.1.1 Blocks

With the help of an Application Programming Interface (API) for python and the *pickle*[1] module, the public blockchain data was obtained and stored in a python object. Only the most important parts of each block were stored in a python object.

For instance, block 400000 has the following properties:

block_id: 400000
The height of the block.

difficulty: 1.63491654909e+11
Determines how difficult it is to find a nonce below the current target number.

generation_amount: 25.0
Current block reward.

generation_fees: 0.33349423
The sum of all transactions included in the block.

hash: 000000000000000004ec466ce4732fe6f1ed1cddc2ed4b328fff5224276e3f6f
The hash of the block header.

merkle_root: b0e8f88d4fb7cbc49ab49a3a43c368550e22a8e9e3e04b15e34240306a53aeec
The hash that contains all transactions of the block.

[1] More information can be found at `https://docs.python.org/2/library/pickle.html`

previous_block: 00000000000000000030034b661aed920a9bdf6bbfa6d2e7a021f78481882fa39
The hash of the previous block header.

next_block: 00000000000000000005421b1b2ee6d06d037557d7f5ec96852542413cfed40c22
The hash of the next block header.

no_transactions: 1660
The number of transactions included in the block.

nonce: 657220870
The nonce that verifies the proof-of-work.

time: 2016-02-25 16:24:44
Time stamp of the block.

total_btc: 26'296.01104117 BTC
The total amount of bitcoins traded in the block.

transactions: An array with all the transactions of the block. For each transaction the hash, fee, inputs, and outputs are included.

short_transactions: A shortened version of the transaction array above, where inputs and outputs belonging to the same users are settled.

The transaction and short_transaction array make use of some special heuristics. A *user* identification (ID) rather than the addresses is stored in those input and output fields. To map the addresses stored in the blockchain to actual users, the two heuristics provided in Meiklejohn et al. (2013) were applied. If a transaction includes multiple inputs, the first heuristic collapsed all the input addresses to one user. As shown in Section 1.3.2, the sender of a transaction must provide the private keys for all the inputs. Even though there are some rare cases where multiple users have to sign their inputs, this assumption should hold in most cases. The second heuristic targets the change address. In this case, every transaction was examined, and whenever certain requirements were met on the input side (e.g., the number of inputs was exactly one) as well as on the output side, the change address belongs to the same user as the input address. As a side note, this heuristic was created with the highest level of certainty possible at the cost of losing some potential address-user relationships. This is because only a small mistake in the heuristic can change the whole user network graph in a fatal way because there are so many dependencies between users in the network.

On its own, this does not reveal the identity of the users; it only connects different addresses to the same user. However, large institutions accepting Bitcoin payments have personal data, which links to an address, available. Even the solely passive analysis of Reid and Harrigan (2011) has been able to unearth much external (meaning outside of the Bitcoin universe) information about the previously identified users. For this reason, Reid and Harrigan have assumed that any large institution practicing

active analysis is able to track and identify large parts of the user network. These findings seem to contradict the original idea behind Bitcoin anonymity. The result is that anonymity in the Bitcoin network is possible as long as the user does not interact with the 'real' world. Once he/she does, it is most likely possible to reconstruct much of his/her transaction history (Reid and Harrigan 2011).

2.2 Methodology

This chapter reviews the methods applied, why the author made particular decisions, and how he implemented them. First, it is necessary to provide an overview of the software that was used. Because all of the blocks are stored as python objects, the main work was done in python 2.7.15. This program could have also been used for creating plots, but the author decided against it because of the grammar of graphics implementation of ggplot2, which is a package for the R programming language and was used to create all the plots in this thesis. To combine python and ggplot, the author typically created a comma-separated values (csv) file at the end of every python script, so he could easily load the file into R, do some adjustments wherever needed, and create the plots. All statistical calculations were performed in R, not in python. This way the author was able to examine the data visually before completing the calculations. The figures included in Chapter 1 were designed with yed,[2] which is a free, platform-independent vector creation tool. It was important that the graphics used in the thesis be vector-based in order to provide a sharp result in the end. To keep the files in sync the author used GitLab with its git implementation, making all of the code accessible at `https://gitlab.business.uzh.ch/dbugma/ba-thesis`. The author set Bitcoin block 500,000 as a general upper bound for this thesis in order to create a homogeneous time scale for all the plots, allowing the results within the subsections to be easily compared. To calculate the correlation between mining concentration and wealth accumulation, for instance, the two datasets must be of equal length. The following chapters are organised in the same three-part manner as the results.

2.2.1 Mining Concentration

In the dataset used (Section 2.1.1), neither the block itself nor the first transaction in each block includes any information about the miner except the address. To create a measure for mining concentration it was necessary to obtain some information about the mining pool involved. For this reason a new dataset was created by first storing the mining pool of each genesis transaction. In the first step this data was fetched via the blocktrail API and stored in a SQLite database. For the sake of convenience, this file was then converted into a simple csv file. Because only a fraction of the complete blockchain data was required, a simple csv file completed the task without any problems.

As is evident, there are many mining pools, and most of them are not really of a relevant size. Figure 2.1 indicates that most of the time, the top five mining pools hold most of the mining power.

[2] `https://www.yworks.com/products/yed`

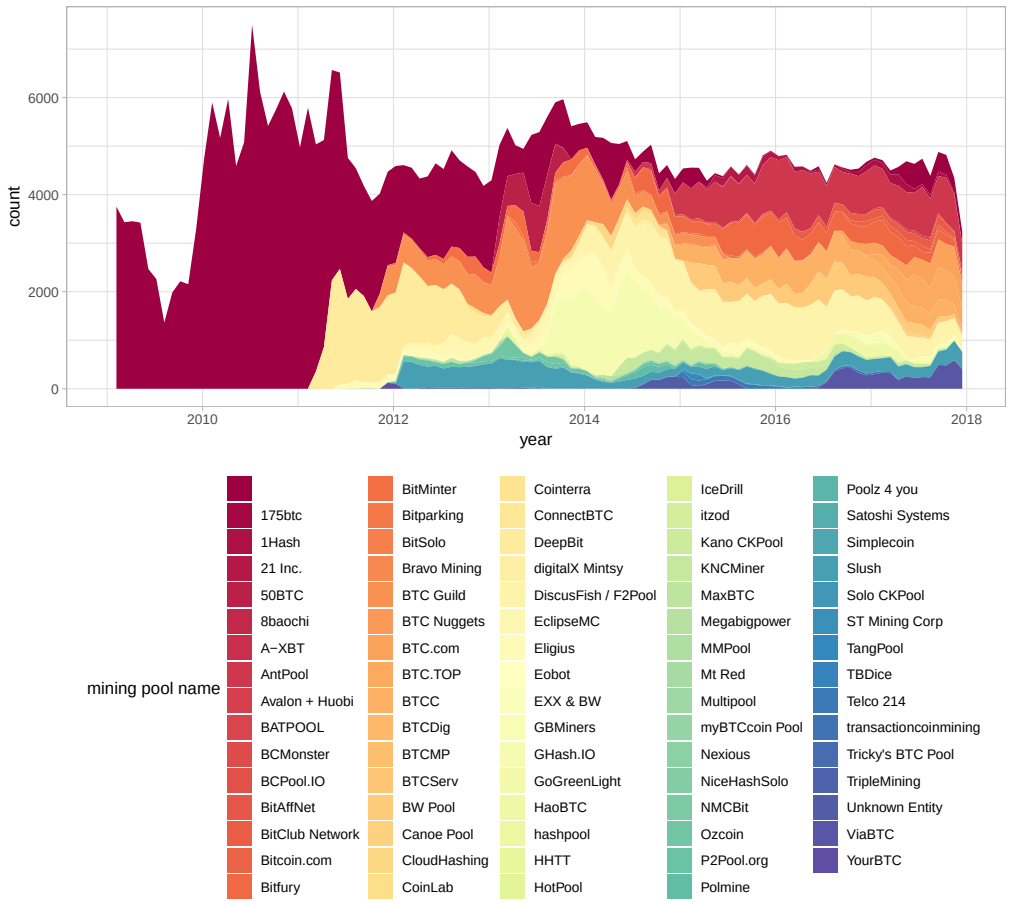


Figure 2.1 – Development of all the mining pools

Examining some other levels and comparing them to the five largest mining pools could be very interesting in case they differ significantly. Also, having multiple measures for mining concentrations offers more data to be compared to the wealth section of this thesis, which is why this thesis examines the three, four, and five largest mining pools. The mining concentration is defined as the share of the blocks mined by the largest mining pools compared to the total blocks. This is a very simple yet powerful way of measuring centralisation. This thesis also examines two other variables, namely 'no mining pool' and 'other mining pools'. Adding these variables makes it possible to see some shifting between all three variables. If the mining concentration of the five largest pools decreases for a given time interval, it is possible to see which of the other variables gained shares. To obtain a precise yet smooth representation of the mining concentration, a time interval of one month was chosen. As a little overview, after the calculations there were three variables for each month:

1. Mining pool concentration (three, four, and five largest pools)
2. Share of remaining mining pools
3. Percentage of blocks where no mining pool was involved

Both an empty field and 'Unknown Entity' (see Figure 2.1) in the mining pool field are classified as 'no mining pool'. Within a python script, the above variables are calculated each month. The created script makes it possible to easily change the number of mining pools to be aggregated. This provides flexibility to change the variables included in the present definition of mining concentration. The results were then plotted and are shown in Figure 3.1.

2.2.2 Miner Identification

The difficulty with finding miners is that the outputs of the genesis transaction (see Section 1.3.4) are often not the miners themselves. As Figure 2.2 indicates, most of the time the output is one single address. In this figure some values bigger than five were ignored (about 14,000 in total) in order to make the plot more readable. Those 14,000 figures are arguably the only cases where the actual mines are found directly in the genesis transaction output. All the remaining outputs have lengths ≤ 5. Having examined the development of mining pools in Section 2.2.1, it is clear that the concentration of mining pools has risen over the years. The outputs of the genesis transactions, however, remain more or less unchanged. This means that mining pools most likely use a single address to receive their block reward and then distribute their reward to their miners later on. It is interesting to note that in 2017 the number of genesis transactions with output length 2 is significantly higher than in the other years. In the applicable cases the second output was simply an address with an output of 0 BTC, which was not decodable.[3] As a result, these cases also have only one real output.

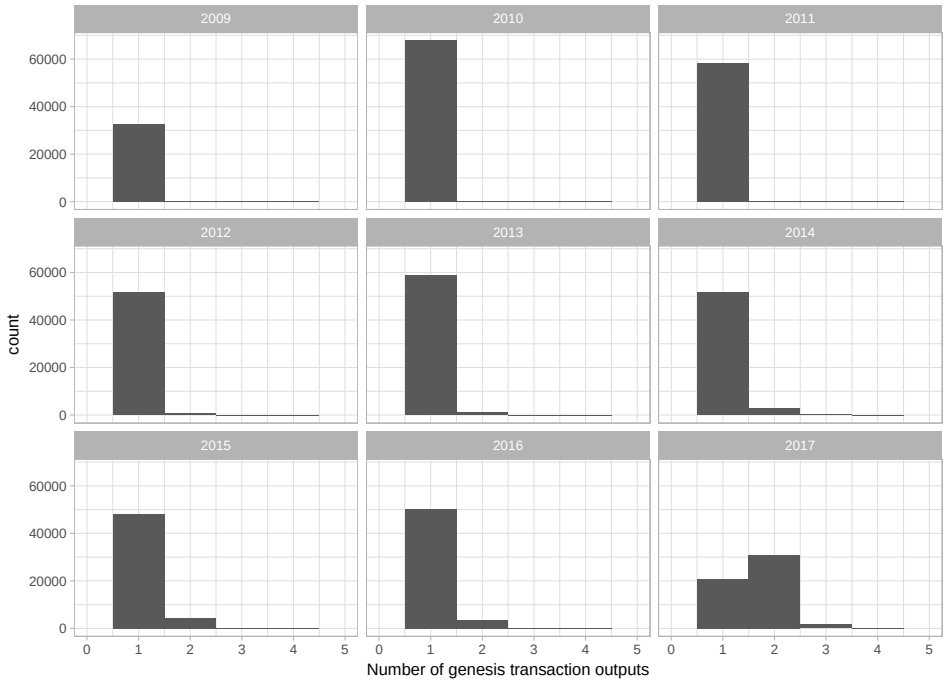


Figure 2.2 – Outputs of genesis transactions

[3]Block 450002 is an example of such a transaction.

Because the output of most genesis transactions only contains a single address, this author went one step further: whenever a genesis transaction output was used as a transaction input in a later transaction, it is referred as a redemption transaction. It was expected that the miners would be obtained as outputs of this transaction. With most of the genesis transaction being mined by a mining pool, this author expected the number of outputs of the redeeming transactions to be a lot higher than the genesis transaction outputs.

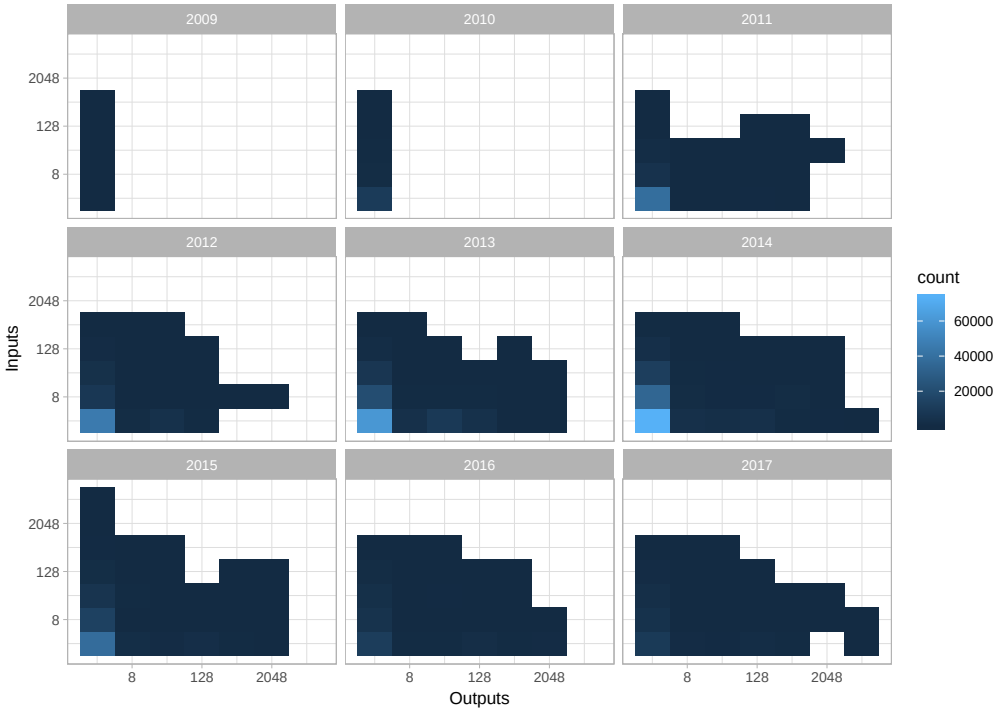


Figure 2.3 – Inputs and outputs of redeeming transactions in log scale

As Figure 2.3 reveals, the aforementioned expectation was not confirmed. Most of the outputs were four or less. This observation makes it clear that those output addresses are very unlikely to be miner addresses. Nevertheless, like in the genesis transaction, there are some exceptions. In some cases, the outputs go up to over 2,000, and even though it is a log scale, the outputs are more evenly distributed than the genesis transaction outputs. With so many outputs being ≤ 4, it seems that the redemption transactions are often just stepping-stones on the way to the miners. It is also apparent that there are often multiple inputs in a redeeming transaction. This is the result of the following behaviour: the large mining pools mine more than one block per day and pay out to their miners once a day (see Table 2.1). The bundling of the rewards into a single transaction leads to multiple inputs because of the way a Bitcoin transaction works. This is more efficient for the pools because it saves them transaction fees.

Even though it is possible to identify certain miners directly through the genesis transaction or through the redemption transaction, those two possibilities of a mining pool to distribute their rewards

are not enough to identify the miners in most cases, which made another option necessary.

When trying to understand some patterns of transaction flows, it is very helpful to make use of an online blockchain explorer such as blockchain.info.[4] Looking at several transactions and following their flow, this author devised the following assumptions:

1. Mining pool receives genesis output to Address 1.
2. Mining pool forwards genesis transaction output to Address 2. This is the redemption transaction and its output is short, which matches the findings expressed in Figure 2.3.
3. Mining pool distributes mining rewards among its miners (when threshold reached). This is the redemption transaction of the redemption transaction (redemption^2).

This author's hypothesis to why a mining pool would chose this redemption strategy consists of two parts. First, by not paying out directly to their miners, mining pools can better manage both transaction fees and general fees. This facilitates the redemption process for mining pool operators by not requiring them to forecast transaction fees. By sending the entire block reward to a single address first, transaction fees and general fees can, if wanted, be easily kept by the pool operator. Secondly, the mining pool wishes to have a second address for privacy reasons. As mentioned in Chapter 1, in order to spend a transaction, one must possess the private key. The more often a certain address is used to create a transaction, the easier it becomes for an attacker with a quantum computer to determine the corresponding private key. In order to avoid changing one's genesis transaction every time, a second address is useful for diversifying risks.

Using this approach, a new dataset was created: The redeem^2 transactions. It is very similar to the redeem transaction dataset previously examined. This author iterated over all the transactions stored in the blockchain with the help of the blocktrail API client. Whenever an output of a redemption transaction was spent later on, that transaction was added to the redeem^2 dataset. It therefore stores the redemption transactions of the redemption transactions.

This author assumed that every genesis transaction is redeemed in a single transaction. Once this transaction was found, the author did not look for any further transactions for the corresponding genesis transaction. This simplifies the analysis and is a reasonable assumption. Even if a redemption transaction had more than one output, only one of those has a redemption^2 output length that would be considered an actual redeeming. This assumption is by no means perfect, and in some cases some miners might be lost, but the effect should be minimal.

Having all the necessary data, this author began the miner identification process. For this purpose, four main cases were differentiated between in keeping with the previous findings. The cases were run one after the other, which made it possible to adjust the code of Case 4, for instance, without

[4] `https://www.blockchain.com/explorer`

having to run Cases 1–3 again. Also, this made the code a lot clearer and easier to understand. The downside is the loss in efficiency, but since time is not really an issue, this is neglectable.

Case 1 is simply when a mining pool directly paid out to more than 10 addresses in the genesis transaction. All the users that correspond to the addresses of such output are identified as miners. In such a case the genesis transaction is considered redeemed and thus no longer relevant.

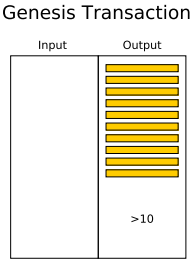


Figure 2.4 – Case 1

In order to obtain realistic results from the next cases, it was important to decide on one parameter first. An important parameter for identifying the miners is the time to redeem, which must first be defined. Time to redeem measures the following: for a given genesis transaction output, how long does it take for it to occur in a redemption transaction again? If this time is too long, the transaction cannot be considered an actual redeeming, and thus its outputs are not the miners. To get a rough idea, this author examined the redeem dataset and the corresponding genesis transactions.

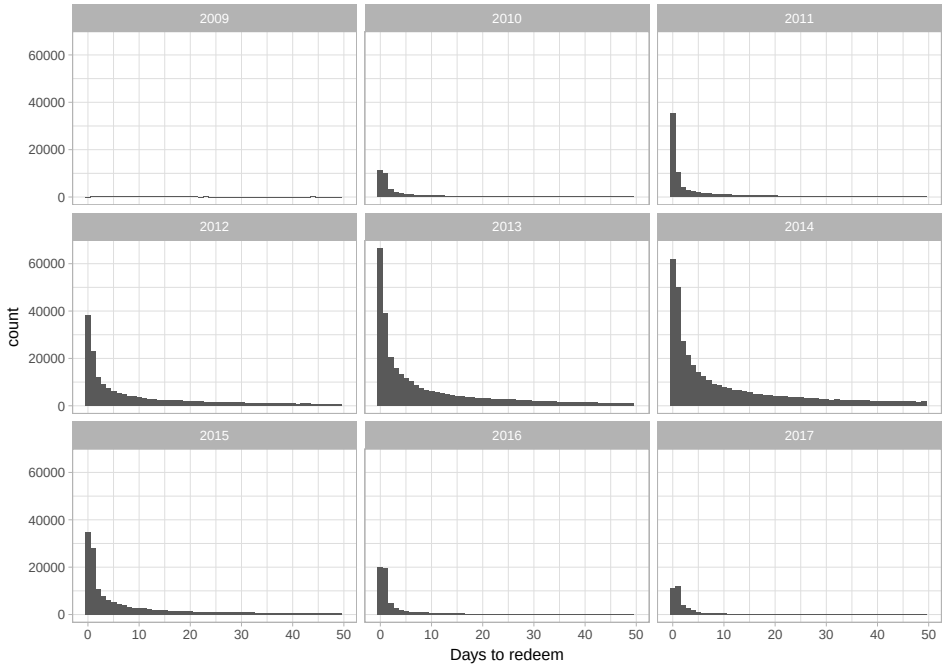


Figure 2.5 – Days to redeem genesis transactions

As shown in Figure 2.5, the time to redeem is very brief, with most of the redeeming taking place within one or two days. The data used to create Figure 2.5 also contains genesis transactions where no mining pool was involved. A single miner is not obliged to pay out within a given timeframe, and this author therefore expected the redemption time for mining pools to be slightly quicker.

In addition, this author also decided to take a closer look at the payout policy of the largest current mining pools (as of 28 August 2018). This only provided a reference point, because it does not take into consideration the payout schemes of older mining pools. Assuming that because of increasing concentration and competition of mining pools payout intervals have become shorter over time, a reasonable safety margin was applied when defining the required parameter.

Name	Market share (blockchain.com 2018)	Payout threshold	Payout interval
BTC.com (btc.com 2018a)	23.95%	NA	daily
AntPool (antpool.com 2018)	14.27%	0.001 BTC	daily
SlushPool (slushpool.com 2018)	11.05%	0.001 BTC	every hour
ViaBTC (viabtc.com 2018)	10.37%	0.01 BTC	daily
BTC.TOP (private)	9.44%	private	private
F2Pool (f2pool.com 2018)	8.03%	0.005 BTC	daily

Table 2.1 – Overview of payout policies of the largest current mining pools

The fact that so-called payout thresholds exist did not affect the decision to set the time to redeem. As previously noted, the mining pools tend to redeem their genesis transaction reward within one day, so the individual miners do not play a role in deciding whether or not the transaction is an actual redeeming. With the payout threshold in mind, however, this implies that not every miner receives his/her shares every day. For a miner with a high level hardware such as the *S9i-14.5 TH/s with PSU* it takes roughly two days[5] to get to the 0.001 BTC threshold, so this miner would not appear in every redemption transaction of the mining pool.

All things considered, this author decided to use a time to redeem of five days for the purposes of miner identification and further analysis.

Case 2 examines the redemption transactions. Whenever the output of a redemption transaction is ≥ 20, all of the inputs are checked to see if they have a previous output that is stored in the genesis transaction dictionary. If so, and if the output length is less then two and the time to redeem was ≤ 5 days, then the outputs of the redemption transaction are miners of this genesis transaction. The genesis transaction is then considered redeemed.[6]

[5] According to cryptocompare.com (2018).

[6] This author originally implemented the program to use output of genesis transaction == 1 but changed it to two with respect to Figure 2.2 (2017). Because the second output is an unspent one with 0 BTC, there is no possibility of finding the 'wrong' redemption transaction, deleting it from the dictionary, and never finding the real miners.

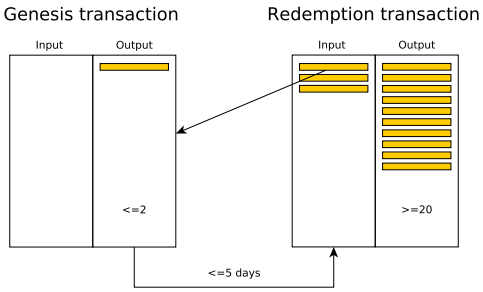


Figure 2.6 – Case 2

Case 3 goes one step further: if the output length of a redemption^2 transaction is ≥ 20, the redemption transaction is checked to see if the output length is < 20. This ensures that the transaction has not already been redeemed in Case 2. Finally, the redeem^2 transaction is checked to see if the corresponding genesis transaction has not yet been redeemed and if the time to redeem is no longer than five days. If all these requirements are met, the outputs of the redemption^2 transactions are the miners' addresses.

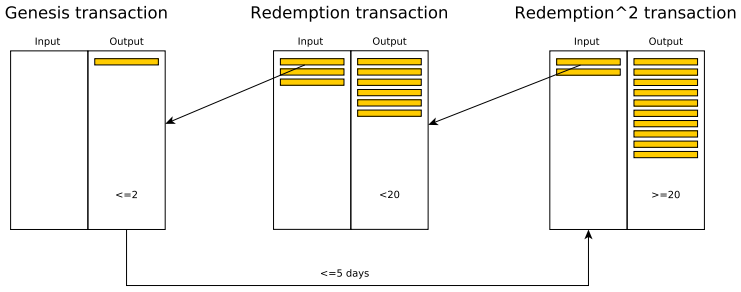


Figure 2.7 – Case 3

Even after these three cases, a lot of the genesis transactions had not yet been redeemed. Many of these transactions turned out to be old genesis transactions from back when mining pools did not exist. A genesis transaction was included in Case 4 when there was no mining pools involved and the transaction was not identified with the help of Cases 1-3. This author originally differentiated between mining pools' and non-mining pools' genesis transactions before Case 1. However, some redemption transactions have outputs larger than 20 even though the genesis transaction was not mined by a pool. The author would still classify the outputs of such redemption transactions as miners. In these cases, without moving this distinction to the end, the correct miners would not have been found.

Case 5 includes the remaining genesis transactions that did not fulfil any of the above criteria. These are all blocks mined by pool that either have a genesis output length between 3 and 10, never produced

a redemption output length exceeding 19, or a time to redeem of above five days was found.

Note that the sequence in which the cases were run is very important. Assume that the genesis output length is 200 and that one of those transactions is later redeemed with an output length of 100. For example, if the miners had not already been identified in the genesis output, the redemption output, which is most likely that large because of a tumbler service, would have been taken as the miners.

In the process of identifying the miners, this author created a vector with values of one for all the miners and zero for the non-miners. The length of this vector matches the inflow, outflow, and balance vectors used in the next part. This means that a simple vector multiplication provides the inflow, outflow, and balance vectors for the miners only. In addition to identifying the miners, this author created a large csv file with more information about each block. This information includes all the miners of the block, shares of the outcome for each miner (address as well as user ID), total transaction output, and much more. The idea of such file is to use it for later analysis: it allows for easy results checking, because just checking the vector does not, for example, show where a particular miner was identified.

2.2.3 Wealth Accumulation

Median

Going through every single transaction in the dataset, two vectors were created: $inflow[i]$ and $outflow[i]$. Those vectors contain the aggregated inflow, respectively outflow of the user i. This author then calculated the balance by simply subtracting the outflow from the inflow. Multiplying the miner vector with the wealth vectors every month resulted in a vector with zeros for all the non-miners and an unchanged value for the miners. In order to have just the miners' wealth left, all the zeros were deleted. In an initial step, this author examined the medians of the balance vector. Because there are so many different possible scenarios where completely different inflow and outflow vectors result in the same balance vector, this author has also examined those two vectors.

Share of Wealthiest Miners

For an initial measure for inequality, this author chose a very simple yet meaningful instrument. Every month the sum of the entire balance vector was calculated. The sum of the largest 1%, 5%, 10%, 20%, and 50% of the values was then divided by the total sum. The result for the largest 1% could then be interpreted as follows: the wealthiest 1% of all miners control x% of the total miners' wealth in the economy, where x is the share of the wealthiest miners.

Gini index

The foundation for the Gini index, also called the Gini coefficient, builds the Lorenz curve. The Lorenz curve follows the same logic as the just-covered shares of the wealthiest users, but rather then only looking at certain levels, it creates a steady line for all possible different levels. Considering wealth, Figure 2.8 provides an example and can be interpreted as follows: 40% of the population controls 5% of the entire wealth, or 80% controls 20% of the total wealth, and so on. However, if one wishes to compare two curves (for different dates, for example) and those curves intersect, it is not possible to say whether the inequality has risen or fallen (Cowell 2011). This is where the Gini index becomes useful.

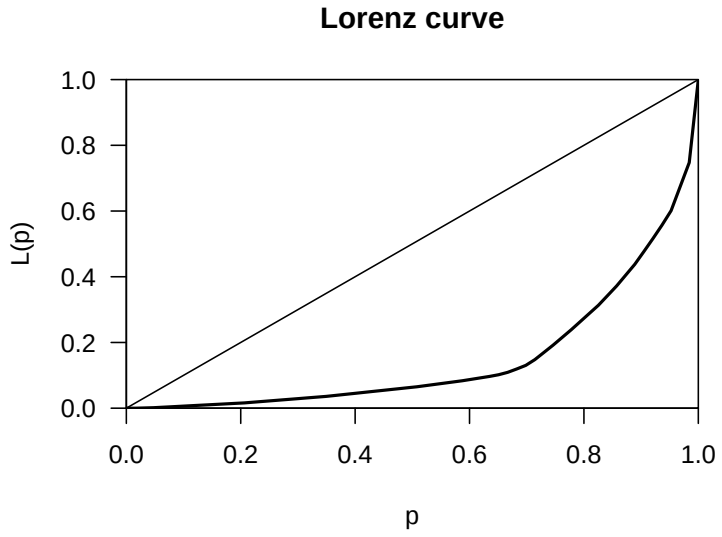


Figure 2.8 – Example of a Lorenz curve

The Gini index calculates the area between the two curves. The diagonal line is simply the reference for complete equality (50% control 50%, and so on). The result is then normalised so that the Gini index is a number in $[0, 1]$. If it is zero, this means that the two lines are identical, and therefore perfect equality is found. If the result is one, there is complete inequality. The benefit of such an index is that one single number provides information about the inequality. The fact that it is normalised also makes it possible to easily compare two different results. There are many different ways to calculate or estimate the Gini index. This author chose the formula from Cowell (2011):

$$G_y = \frac{1}{2n^2\bar{y}} \sum_{i=1}^{n} \sum_{j=1}^{n} |y_i - y_j| \tag{2.1}$$

Despite being widely used, the Gini index has two main disadvantages. Allison (1978) has stated that because the Gini index is not decomposable, it is not possible to calculate inequality of or within

subgroups. Cowell (2011) has mentioned that 'the Gini coefficient can only be decomposed if the constituent subgroups are "non-overlapping" in the sense that they can be strictly ordered by income levels' (p. 162). Since this thesis does not deal with decomposing and subgroups, this disadvantage is not presently relevant.

The other possible issue is that Gini delivers slightly different results if changes are made in different places within the vector. If two users located in the middle of a distribution transfer a certain value, this transaction has a much greater impact on the Gini index in comparison with the two users being on either tail of the distribution. For instance, User x has \$10,000 and transfers \$100 to a user with \$9,000. This transaction reduces the Gini index more significantly than a \$100 transfer between a user with \$100,000 and \$99,000 or \$5,000 and \$4,000 (Cowell 2011). 'This valuation may be desirable, but it is not obvious that it is desirable' (Cowell 2011, p. 26). Even thought this assumption makes sense based on this author's personal understanding of inequality, other understandings are conceivable.

In terms of input values for the Gini function, negative inputs may lead to Gini coefficients outside of the desirable interval $[0, 1]$ (Chen, Tsaur, and Rhai 1982). Because the wealth values in this thesis are corrected to be greater than zero, this does not apply in this case. This thesis only considers values greater than zero when calculating the Gini coefficient in order to avoid including non-miners.

Because none of the downsides are of particular importance for this thesis, the simplicity and standardised results of the Gini coefficient returns are satisfactory, and therefore the Gini coefficient is used as the main measure for inequality in this thesis.

2.2.4 Correlation

To compare the inequality results with the mining concentration, two different types of correlation are used, namely *Pearson Correlation Coefficient* ρ and *Kendall Tau Rank Correlation Coefficient* τ. ρ measures the *linear* relationship between two variables and is defined as follows:

$$\rho = \frac{\text{cov}(X, Y)}{\sigma_x \sigma_y} \tag{2.2}$$

$$= \frac{\sum_{i=1}^{n}(x_i - \overline{x})(y_i - \overline{y})}{\sqrt{\sum_{i=1}^{n}(x_i - \overline{x})^2 (y_i - \overline{y})^2}} \tag{2.3}$$

The result is a number in the interval $[-1, 1]$. A value of one exhibits a perfect positive and linear association between the two variables. A positive association means that if one variable increases, the other also does so. As a contrary, a value of -1 indicates a perfect negative linear relationship. If the correlation is zero, there is no linear association between the two variables, though there might be another association, which is why examining scatter plots before calculation correlation can be very beneficial.

In contrast to ρ, τ is a *rank-based* correlation that measures the *monotonic* relationship between two variables. There exist three different versions - τ_a, τ_b, τ_c - the latter of which is not covered in this thesis. τ_a is the simplest and defined as follows:

$$\tau_a = \frac{c-d}{c+d} = \frac{S}{\binom{n}{2}} = \frac{2S}{n(n-1)} \tag{2.4}$$

where $c =$ the number of concordant pairs and $d =$ the number of discordant pairs.

Assume there are n data pairs (x, y) in a data set. These pairs are then sorted by x in ascending order. The first pair (x_1, y_2) is compared with all the other pairs $(x_2, y_2), \ldots, (x_n, y_n)$. Because the pairs were sorted in ascending order it holds that $x_1 \leq x_2 \leq x_n$.

It is now necessary to compare (x_i, y_i) to (x_j, y_j) with $i = 1, \ldots, n-1$ and $j = i+1, \ldots, n$ and check for each pairing (total of $n(n-1)/2$ pairings):

- if $x_i < x_j \wedge y_i < y_j \implies$ *concordant*
- if $x_i < x_j \wedge y_i > y_j \implies$ *discordant*

This procedure works unless one encounters ties, because then $x_i = x_j$ or $y_i = y_j$ yielding neither a result for each pair where ties occur nor a normalised result. If ties are present between the two ranked variables, τ_b is required, and the following equation should be used instead:

$$\tau_b = \frac{S}{\sqrt{n(n-1)/2 - T}\sqrt{n(n-1)/2 - U}} \tag{2.5}$$

$$T = \sum_t t(t-1)/2 \tag{2.6}$$

$$U = \sum_u u(u-1)/2 \tag{2.7}$$

where $t =$ number of observations of variable x that are tied and $u =$ number of observations of variable y that are tied.

The calculation mechanism is very similar to the first case; it just adds two more possible results for each pairing:

- if $x_i \neq x_j \wedge y_i = y_j \implies$ tied y
- if $x_i = x_j \wedge y_i \neq y_j \implies$ tied x

These adjustments are required in order to receive a value of $[-1, 1]$ again if ties occur. There were many ties for certain values (i.e., 50 BTC) within the wealth vectors in this thesis; therefore, τ_b was used for further analysis.

Chapter 3

Results and Discussion

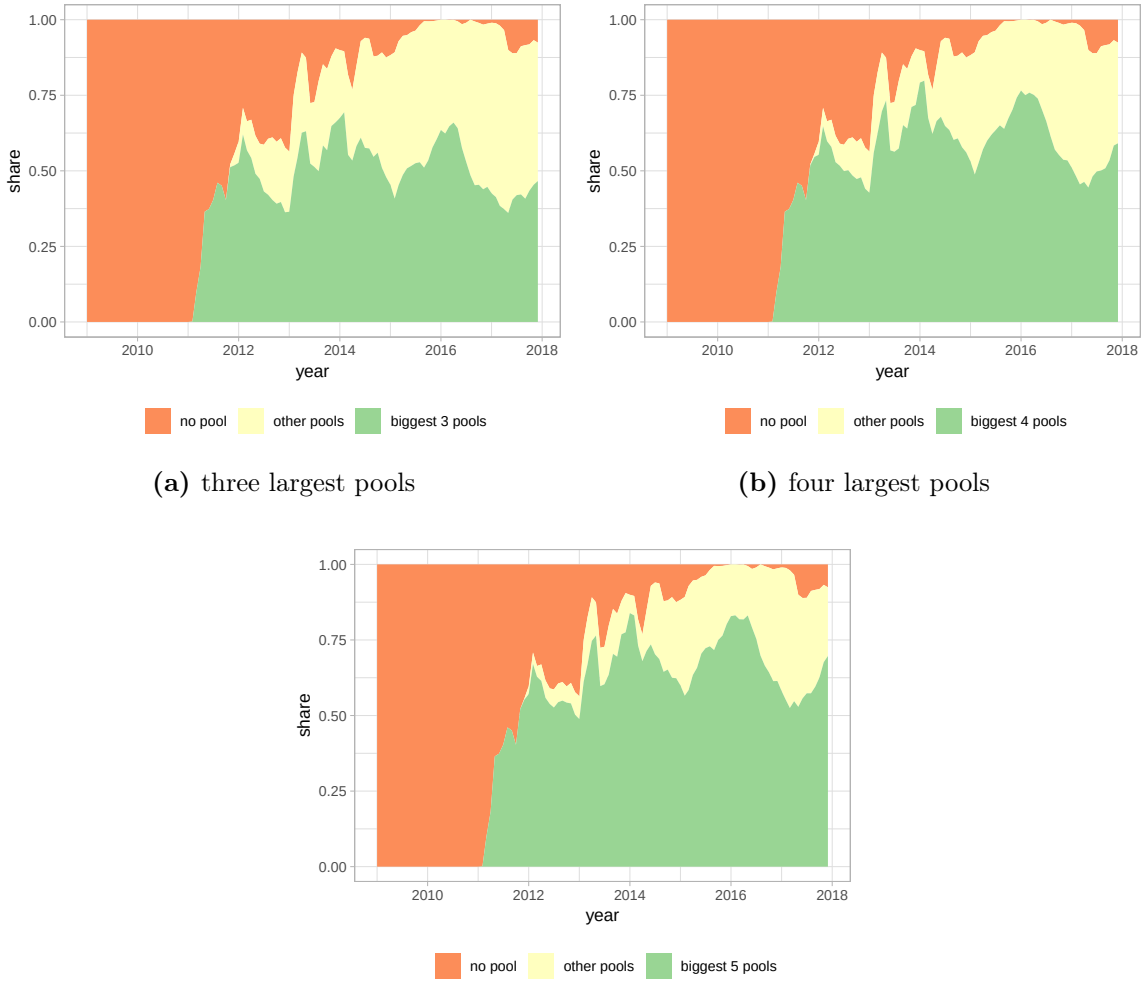
3.1 Mining Concentration

Mining concentration has developed very similarly among the three different definitions. From the year 2012 onward all three mining concentrations are moving roughly within a 35% spread and with the same ups and downs. The main difference is just the overall level. Here they also increase in a similar manner: the average concentration increases from the top three to the top four by 8.6% and from the top four to top five by 6.2%. It is clear that the increments shrink with each additional pool because by definition a smaller pool is added each time. The main result here is that the top three mining pools averaged 51.1% of all mined blocks from January 2012 to December 2017 and could, as a result, control the network. In addition, the five largest mining pools control almost two thirds of the network. Compared to other cryptocurrency mining entities, these findings are quite positive. In Ethereum, for instance, the three largest pools mined 61% of the total amount of weekly blocks for 10 months beginning 15 July 2016 (Gencer et al. 2018).

With respect to Figure 3.1, mining by pools has become much more popular over the years, with a slight crash at the end of 2017 where 'no pools' have gained shares again. Of particular interest are the rather large spreads in the concentration. Because the 'other pools' do not seem to react so drastically to the decrease in mining concentration, this author has made the following assumptions: every large decrease in mining concentration is due to a new mining pool entering the market or a pool improving the conditions. As a reaction, many miners switched mining pools and the concentration of the largest pools therefore decreased. As soon as the new/updated pool made it into the largest mining pools, the mining concentration increased again. This is very desirable behaviour because it indicates that miners are not limited to a specific mining pool and that real competition exists.

It should be noted that the details used about the mining pools were stored in the blockchain by the miners themselves, so even though it might say 'Unknown' in the mining pool field, this might not be true. Also, the mining pools voluntarily publish their data in order to demonstrate their power.

Because it is desirable for a pool to have the largest possible mining power, some pools might act as though they use the same domain, which would lead to finding more exaggerated centralisation than there really is. As long as the centralisation found is this thesis is not smaller than in reality, this author does not rate this as problematic.



(a) three largest pools

(b) four largest pools

(c) five largest pools

Figure 3.1 – Monthly mining concentration of the largest pools

3.2 Miner Identification

Figure 3.2 illustrates the development of the genesis transactions and the related case, which found the miners of such transactions. Out of the total of 500,000 blocks, the miners of 327,048 ($\approx 65.4\%$) were identified. Sadly, the methods applied failed to find the miners in roughly every third block. Fortunately, those blocks are equally distributed among the years and should, as a consequence, not really manipulate the results. The spike at the end is most likely because those blocks have not yet

had a chance to be redeemed. Those blocks were either not redeemed in time or their output lengths were never large enough. Those blocks are often transactions to single outputs, and those outputs are then also spent to a single output, and no matter how long one follows that path, there seems to be no end in sight. This author cannot offer a plausible explanation of this behaviour except that those are private mining pools that do not have to pay out to their miners.

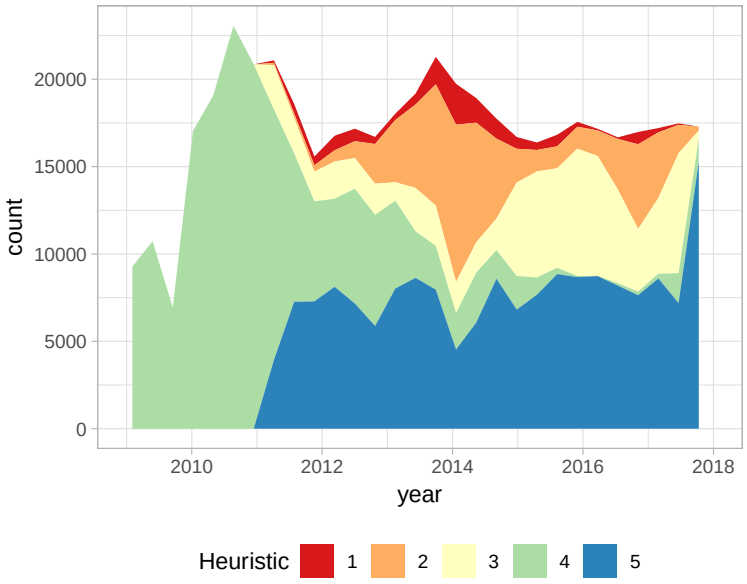


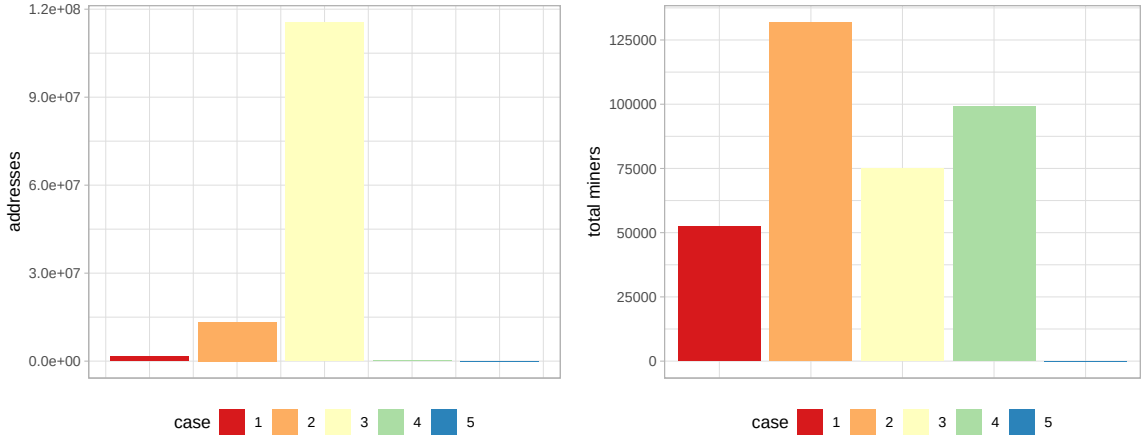
Figure 3.2 – Genesis transactions redeemed by case

This author did not attempt to identify any miners belonging to Case 5 because having a smaller dataset is preferable to including more miners with less certainty about the correctness.

As expected, there are only a few transactions where Case 1 could identify the miners directly through the genesis transaction. Interestingly, however, the total number of miners identified by Case 1 is much closer to the other cases than expected (see Figure 3.3b). Pools paying out to their miners directly must have huge genesis outputs as a result. Therefore, even though those blocks are very rare, they can under no circumstances be neglected. The introduction of Case 3 and the redemption^2 dataset seem to have taken over Case 2 lately, and thus its more complex implementation was definitely worth it. With respect to Figure 3.3, Case 3 identified more miners then Case 1, but Case 2 identified by far the most miners. From an efficiency standpoint it makes a lot more sense to redeem directly to save transaction fees, especially considering the large surge in transaction fees (compared to the block reward) in 2017 (btc.com 2018c). In the early days of Bitcoin when there were no mining pools, Case 4 identified the miners. In this case, the output of the genesis transaction is the miner. With the birth of mining pools, this case drastically lost popularity, but it still occurs every now and then, and it was important to include early day miners in this analysis.

Figure 3.3 displays the enormous number of addresses (130,568,771) belonging to the miners. For all identified blocks this leads to an average output length of approximately 399 per redemption

transaction. This is a much more realistic result than that first obtained with the single redemption transaction data (compare to Figure 2.2 and Figure 2.3, where the length of most outputs was one or two). Also, the effect of the heuristics applied to combine addresses into users are very noticeable since those addresses collapsed to 358,404 miners when examined at a user level. This implies that there are likely a few very large miners that mine a considerable amount of blocks, which results in an almost 1:1 mapping for identified blocks and identified miners. Case 3 clearly found the most addresses, which could be explained in a variety of ways. One such reason is that because Case 3 identifies mostly newer blocks, which have higher complexity and thus require more miners, leading to larger outputs. Furthermore, newer users might renew their addresses more often, which would explain why, relative to the addresses, few miners were identified. A third possibility is that some addresses found with Case 3 were already identified as miners in an earlier case and were therefore not shown again in Case 3.



(a) Miners' addresses **(b)** Miners

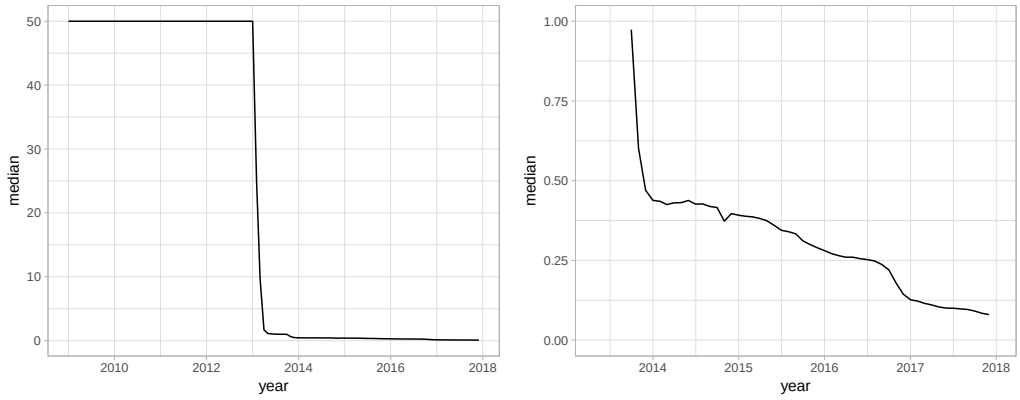
Figure 3.3 – Addresses and miners identified by case

3.3 Capital Accumulation

Before looking into inequality, an independent view of the data using medians is presented. In addition, this section covers not only balances but also inflow and outflow in order to allow for a better understanding of the dataset.

Median

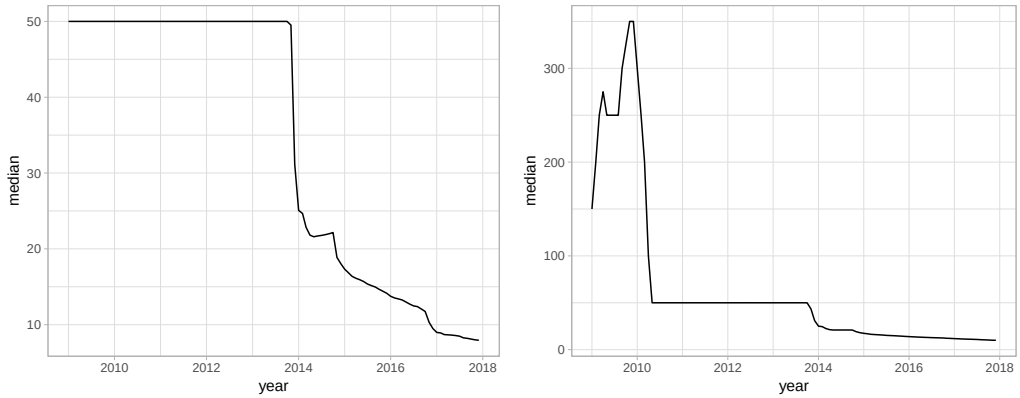
As Figure 3.4 shows, until approximately the year 2013 the median balances of the miners were 50 BTC. This was the mining reward at that time, which makes a lot of sense for the inflow. The balance having the same median means that those inflows were not spent at that time. This author does not assume that they were spent after that, rather the amount of miners has drastically increased



(a) whole dataset **(b)** starting from May 2013

Figure 3.4 – Median of balance

beginning in 2012 (see Figure 3.1) because of mining pools. The increase in mining users was not immediately visible in the medians, of course, because the new miners have much lower balances, and therefore the median is only affected when the mining vector becomes approximately double the size.[1] Thus, the medians shifted in 2013 and no longer include those early days miners. For this reason the balance from 01/05/2013 is included in a separate figure so that the development of the newer miners' median can be seen more clearly.



(a) Inflow **(b)** Outflow

Figure 3.5 – Median of inflow and outflow

It should be noted that those figures were created based solely on values larger than zero and thus only the cases where anything was spent at all. The inflow is not surprising because it is natural for the input of miners to fall due to the halving of genesis rewards and for the same reason stated in the balance section: if there are more miners with smaller values, the median shrinks. As far as the outflows are concerned, they behave in a similar manner, but the overall level is a bit lower (except

[1] Assuming all the values in the vector are equal.

for the early years). On a user basis, it is clear that the outflow cannot be higher than the inflow, and it makes sense that the outflow is lower because of savings. Considering medians, this may not be guaranteed, because the two vectors may not include the same users after deleting zeros.

Shares of the Wealthiest Miners

Interestingly, the shares of the five different levels differ significantly (see Figure 3.6). While inequality in balance increased to a similar extent for all levels until 2014, inequality actually decreased for the 1%, 5%, and 10% levels after this point in time. Once again, the increase in mining popularity, which enlarges the user base, is presumably the key factor for this finding. If the absolute amount of the very rich miners stays more or less the same but many other miners enter the system, the share of the very rich decreases as a result. Another possible explanation could be that the wealthier miners began spending large parts of their stored bitcoins because it became a more accepted payment method.

The fact that almost 100% of the total wealth is held by half the miners led to the conclusion that there have to be many very small values in the vector, which is why this author created another figure with a different minimum wealth level (0.01 instead of 0). As seen in Appendix C, the impact of such a change is very minimal, and this author does stick with Figure 3.6, which includes all the values $\neq 0$ and is therefore is more representative.

All in all, this result is surprising, and the reduction in the inequality of the 1%, 5%, and 10% levels can be categorised as a positive development.

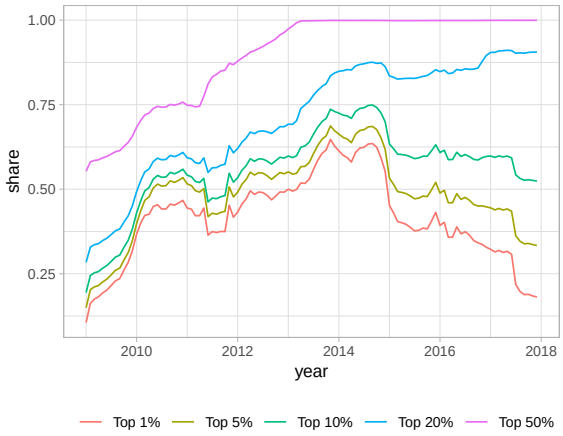
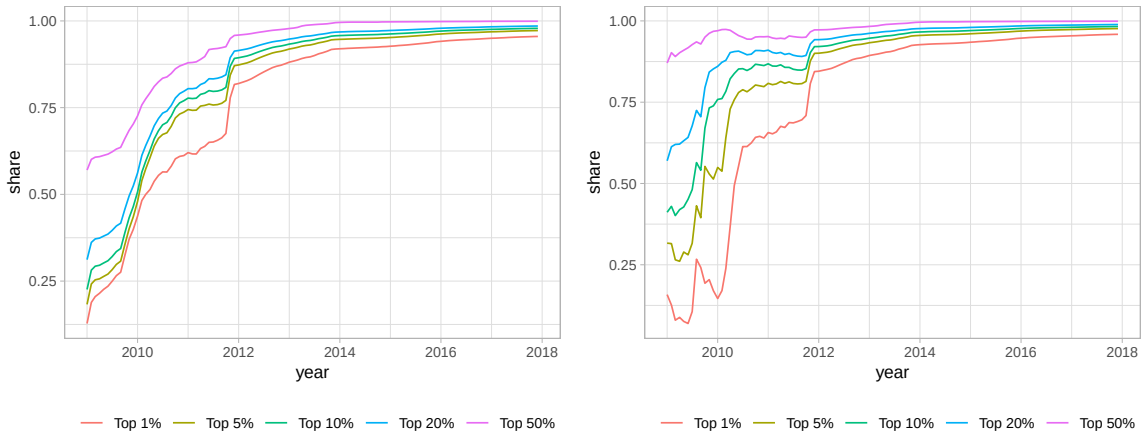


Figure 3.6 – Share of the wealthiest miners (balance)

Looking at the inflow and outflow development, this author found a completely different result for the 1%, 5%, and 10% levels. This figure proves the fact that inflow and outflow are approximately equally unequal. The miners who receive the greatest amount of bitcoins also tend to spend by far the most. The findings illustrated in Figure 3.6 can therefore be explained to a certain extent. It is important to mention that because this figure does not settle the inflow and outflow (in comparison to

the balance plot), an increase in inequality is natural. Accumulated income and outcome vectors are being examined. Because the mining reward of Bitcoin naturally halves after a certain time period, it was easier for miners in the early days to accumulate 50 BTC. Those inflows still remain in the inflow vector, and thus a modern miner would have to mine four times the amount of blocks to have the same total inflow. This effect is somewhat automatically corrected when creating the balance vector because the higher inflows in the early days are settled against comparably higher outflows. This is why the inequality of the balance vector does not quite follow the same patterns.



(a) Inflow **(b)** Outflow

Figure 3.7 – Share of wealthiest miners (inflow and outflow)

Gini index

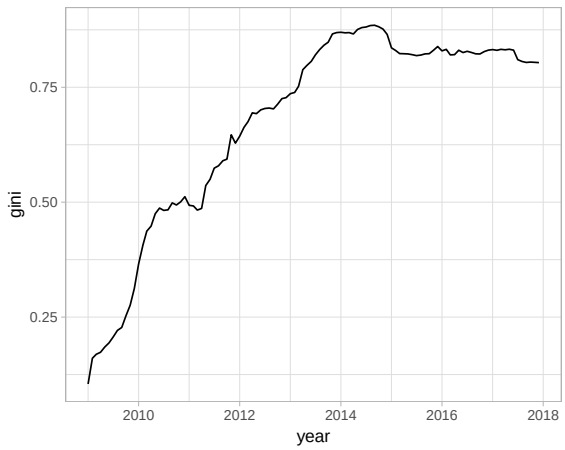
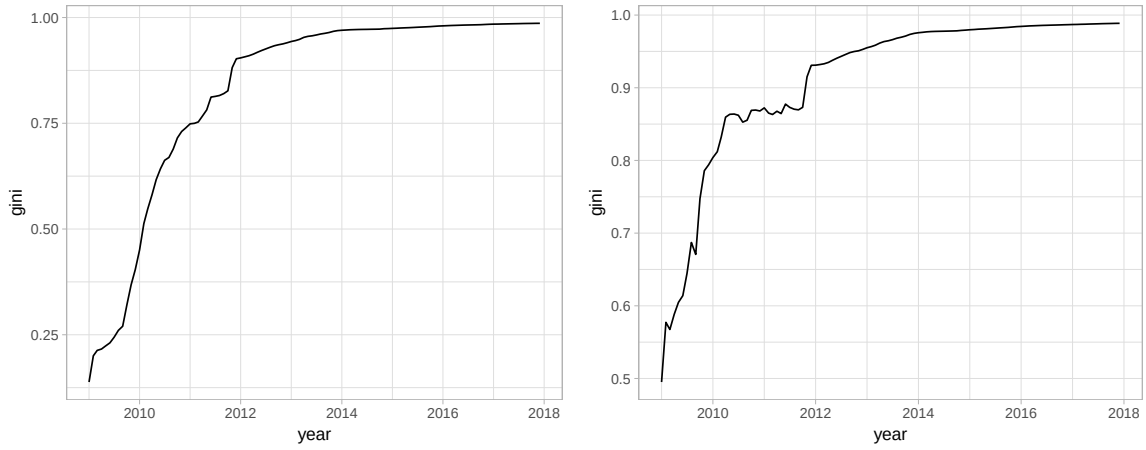


Figure 3.8 – Gini index of balance

The similarities to the previous section are especially noticeable when comparing the Gini of the balances to the top 20% of the balances; nevertheless, the Gini coefficient manages to combine all

the levels from Section 3.3. This results in the Gini index providing clearer evidence of the very large presence of inequality of the balance vector. Regardless, according to the Gini index inequality decreased in the middle of 2014 and has been stable and slightly deceasing ever since.

The Gini index for the inflow and outflow vectors also resembles the results of the previous section. While it is very difficult for newer miners to catch up with the accumulated inflow and outflow of the established miners, the overall increase in equality in those two vectors is very clear. While they are interesting to look at, they are not the main result of this thesis.



(a) Inflow **(b)** Outflow

Figure 3.9 – Gini index of inflow and outflow

While it is always difficult to interpret such results without a point of reference, compared to the real-world Gini coefficient, these results are very high. At 0.651, South Africa has the highest Gini coefficient in the years 2011–2015 – most European countries are in the area of 0.3 (wider.unu.edu 2018). Despite the fact that those are two different worlds, the level of inequality in Bitcoin is tremendously high.

3.4 Combining Mining Concentration and Capital Accumulation

The two results consisting of wealth inequality and mining concentration measures are combined in this section with the help of the Pearson and Kendall's correlation coefficient. Before calculating correlation, the data should first be examined. This prevents using the linear measure on a dataset that fits a different approximation. In Figure 3.10 the scatter plot of the highest correlation found is shown. It is clearly visible that there is a strong linear relationship between the Gini index of wealth balances and the mining concentration of the five largest mining pools.

Table 3.1 summarises all of the correlations. There is a very high correlation between mining concentration and wealth inequality. Interestingly, the correlation increases if more mining pools are covered, so the top five mining pool concentration correlates to the Gini balance with 0.91 (Pearson),

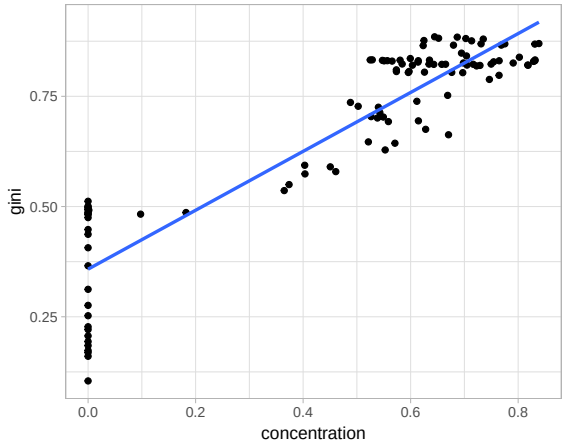


Figure 3.10 – Correlation between mining concentration (largest five mining pools) and Gini index (balance)

which is the highest result obtained. Nevertheless, the slightly lower results for the top three and top four are still considered very highly correlated. All the correlation coefficients are statistically significant on every common significance level α (double-sided t-test, with $H_0 = 0$).

	ρ	τ_b
Top 3	0.88 (0.000)	0.55 (0.000)
Top 4	0.90 (0.000)	0.60 (0.000)
Top 5	0.91 (0.000)	0.62 (0.000)

Table 3.1 – Correlation of mining concentration and Gini (balance)

Controlling for Year

With the help of correlation, associations can be found between two variables, but it does not reveal whether or not there is a causal relationship. In this particular case, one could be tempted to claim that there might be a causal link between mining concentration and inequality. If mining concentration increases, inequality does the same, or the opposite way. There could, however, also be a third variable effecting both of the other variables; such variables are called confounding variables. There would then be no causation between the two variables at all. This author found the very strong correlation a bit suspicious and assumed the correlation to be so strong not because the two variables are linked together but rather because of a third as yet undiscovered variable. This author chose the variable *year*, but any variable directly linked to year – *popularity*, for instance – should lead to similar findings. To gain more clarity, the correlation for each year was corrected in Figure 3.11. The correlation within those years is drastically different.

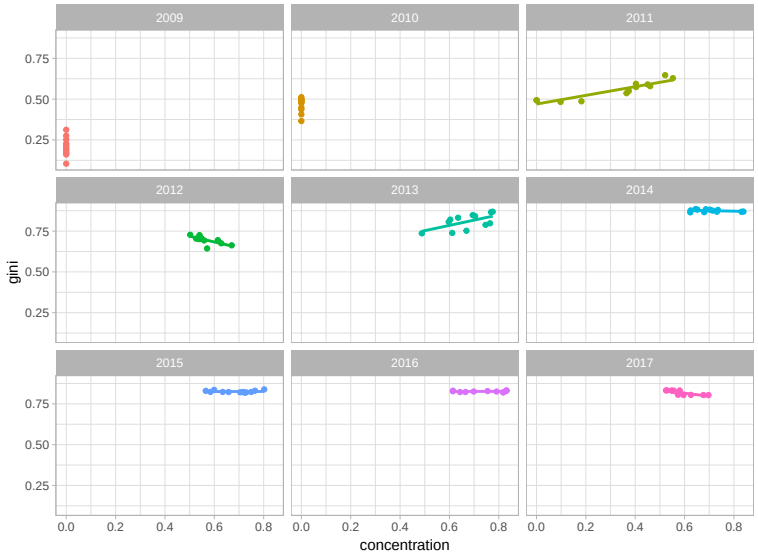


Figure 3.11 – Correlation between mining concentration (largest five mining pools) and Gini index (balance), controlled for year

Table 3.2 includes all the correlations as well as their significance. The visual representation has already suggested that there is no significant correlation in the years 2014, 2015, and 2016. All the other years' (except 2009 and 2010) correlations are significant at $\alpha = 0.05$. They are, however, not invariably positive or negative, and their values are nowhere near the previous ones. Granted, there are not as many data points per year, and the statistical significance should be used with caution. All things considered, the assumption turned out to be true; therefore, large parts of the previously found correlation are simply due to the increase of both variables over time.

Year	ρ	τ_b
2011	0.92 (0.000)	0.70 (0.001)
2012	-0.69 (0.013)	-0.67 (0.002)
2013	0.61 (0.037)	0.52 (0.021)
2014	-0.31 (0.328)	-0.15 (0.545)
2015	0.03 (0.929)	-0.06 (0.841)
2016	0.07 (0.835)	0.15 (0.545)
2017	-0.77 (0.003)	-0.73 (0.000)

Table 3.2 – Correlation of mining concentration (five largest mining pools) and Gini index (balance), controlled for year

Chapter 4

Conclusion

Using the applied methods, this author was able to answer all parts of the research question and provide answers for mining concentration and capital accumulation on a user – namely miner – basis, which has not previously been researched.

Considering mining concentration, the research demonstrated that the three largest mining pools have an average of approximately 50% of the mining power of the network. Thus, if the three largest pools were controlled by a single entity (or possibly even two, if the single entity controlled 33% in total (Eyal and Sirer 2018)), the Bitcoin ecosystem would be in danger. The spread of mining concentration is rather large. As a function of time, from 2012 onward mining concentration moves within a $\approx 35\%$ interval. The data used in the thesis was voluntarily published by the pools and thus might not be entirely accurate.

By implementing four different cases, the miner identification made it possible to identify the miners of about two thirds of the 500,000 blocks covered in this thesis. For the remaining blocks no corresponding redeeming transactions could be found. In further research a different case or an entirely different approach would be required to identify the miners of those blocks. While very few addresses in the outputs of the genesis transactions themselves were found, the methods applied in this thesis resulted in an average output length of 399 for the identified redemption transactions, which is much more realistic than only looking at genesis transactions.

The shares of the top 1%, 5%, and 10% of the wealthiest miners (balance vector) increased rapidly until 2014. After that, they almost fell back to their original levels. The 20% and 50% levels have risen steadily throughout the entire timespan. Similar behaviour was found for the Gini coefficient until 2014. The Gini coefficient, however, has been more stable since 2014 and has stayed at approximately 0.8 for the balance vectors. As a result, it is an accurate summary of the shares of the wealthiest miners. Inflow and outflow vectors do not compensate for inflow with outflow and have therefore continuously risen and demonstrated even higher levels of inequality.

Going up to as high as 0.91, a very strong Pearson correlation ρ between mining concentration and wealth inequality is found. Kendall τ results in smaller coefficients; however, is statistically significant in the same manner. Nevertheless, mining concentration and wealth inequality do not share a causal relationship, since results change when introducing a confounding variable. Controlling for year, correlation is no longer significant or strictly positive for all years. As a conclusion, it is plausible to assume that year, or any other directly year-linked variable, has a causal effect on both mining concentration and inequality.

Some of the key takeaways are that planning a thesis is of extraordinarily importance in a computationally intense field of research. Discarding finished work and starting over again is very time-consuming and at times could have been easily avoided by storing slightly more data than this author expected to require. All in all, this author enjoyed the work and is taking away much more from this work than expected. In addition to all the knowledge related to Bitcoin and cryptography, this work has taught this author a great deal in terms of programming, git, Linux, structuring and illustrating data with R and ggplot, and much more.

Appendices

Appendix A

Pay-to-PubKeyHash Verification Process

<sig>: Add the sender's signature to the stack.

<pubKey>: Add the corresponding public key to the stack.

OP_DUP: Duplicate the last item of the stack (public key).

OP_HASH160: Hash the duplicate twice, first with SHA-256, then with RIPEMD-160. According to Section 1.1.4 this results in the Bitcoin address.

<pubKeyHash>: Add the public key hash of the *referenced* output (the unspent output that is now desired to be spent) to the stack.

OP_EQUALVERIFY: Checks if last 2 stack items are equal (pubKeyHash and pubKey). If this is true, the sender proves that he/she owns the public key to the address in the referenced output.

OP_CHECKSIG: Signature is checked for the last 2 stack items (<sig> and <pubKey>).[1] More precisely, it recreates a copy of the transaction depending on the signature type. It then double-SHA-256-hashes this copy and checks it against the signature (to which <pubKey> was applied). This ensures that the sender possesses the corresponding private key to the public key of the last step. Furthermore, it proves that the transaction was not changed, or at least the parts that were signed.[2]

[1] Depending on the method used for signing the transaction, see section 1.3.2
[2] See section 1.3.2

Appendix B

Signature Types

SIGHASH_ALL: This signs all the inputs and outputs, which is the default.

SIGHASH_NONE: Signs all of the inputs. The outputs are not signed, which allows anyone to change them.

SIGHASH_SINGLE: Only the output that corresponds to this input is signed. This makes sure that nobody can change ones section of the transaction. The other parts however, can be changed by the owner of the required private key/s.

Appendix C

Alternative Plots

(a) wealth > 0 **(b)** wealth > 0.01

Figure C.1 – Differences of balance vector with different minimum wealth level

Bibliography

Allison, Paul D (1978). "Measures of inequality". In: *American sociological review*, pp. 865–880.

antpool.com (2018). *Support*. URL: https://www.antpool.com/support.htm (visited on 10/28/2018).

bitcoin.it (2018a). *Block*. URL: https://en.bitcoinwiki.org/wiki/Block (visited on 09/20/2018).

— (2018b). *Block hashing algorithm*. URL: https://en.bitcoin.it/wiki/Block_hashing_algorithm (visited on 09/23/2018).

— (2018c). *Mining hardware comparison*. URL: https://en.bitcoin.it/wiki/Mining_hardware_comparison (visited on 11/04/2018).

— (2018d). *Non-specialized hardware comparison*. URL: https://en.bitcoin.it/wiki/Non-specialized_hardware_comparison#Nvidia (visited on 11/04/2018).

— (2018e). *Protocol documentation*. URL: https://en.bitcoin.it/wiki/Protocol_documentation#Addresses (visited on 09/10/2018).

— (2018f). *Protocol documentation*. URL: https://en.bitcoin.it/wiki/Protocol_documentation#Merkle_Trees (visited on 10/05/2018).

— (2018g). *Transaction*. URL: https://en.bitcoin.it/wiki/Transaction (visited on 09/23/2018).

bitcoin.org (2018a). *Blockchain Guide*. URL: https://bitcoin.org/en/blockchain-guide#introduction (visited on 11/03/2018).

— (2018b). *Developer guide*. URL: https://bitcoin.org/en/developer-guide#signature-hash-types (visited on 10/29/2018).

— (2018c). *Developer guide*. URL: https://bitcoin.org/en/developer-guide#term-merkle-tree (visited on 09/02/2018).

blockchain.com (2018). *Bitcoin Hashrate Distribution*. URL: https://www.blockchain.com/en/pools (visited on 08/28/2018).

btc.com (2018a). *Payment Confirmation*. URL: https://pool.btc.com/helpCenter?id=payment (visited on 10/28/2018).

— (2018b). *Script types*. URL: https://btc.com/stats/script?time_range=all&latest_block=547750&static_type=amount_stats (visited on 10/29/2018).

— (2018c). *Transaction fees*. URL: https://btc.com/stats/fee (visited on 11/04/2018).

Chen, Chau-Nan, Tien-Wang Tsaur, and Tong-Shieng Rhai (1982). "The Gini Coefficient and Negative Income". In: *Oxford Economic Papers* 34.3, pp. 473–478. ISSN: 00307653, 14643812. URL: http://www.jstor.org/stable/2662589.

Cowell, Frank (2011). *Measuring inequality*. Oxford University Press.

cryptocompare.com (2018). *Mining calculator*. URL: https://www.cryptocompare.com/mining/calculator (visited on 10/28/2018).

Eyal, Ittay and Emin Gün Sirer (2018). "Majority is not enough: Bitcoin mining is vulnerable". In: *Communications of the ACM* 61.7, pp. 95–102.

f2pool.com (2018). *Help*. URL: https://www.f2pool.com/help/? (visited on 10/28/2018).

Franco, Pedro (2015). *Understanding Bitcoin : cryptography, engineering and economics*. eng. Wiley finance series. Chichester: Wiley. ISBN: 978-1-119-01915-2.

Gencer, Adem Efe et al. (2018). "Decentralization in bitcoin and ethereum networks". In: *arXiv preprint arXiv:1801.03998*.

Gervais, Arthur et al. (2014). "Is bitcoin a decentralized currency?" In: *IEEE security & privacy* 12.3, pp. 54–60.

Johnson, Don, Alfred Menezes, and Scott Vanstone (2001). "The elliptic curve digital signature algorithm (ECDSA)". In: *International journal of information security* 1.1, pp. 36–63.

Kelly, Morgan (2000). "Inequality and crime". In: *Review of economics and Statistics* 82.4, pp. 530–539.

Kondor, Dániel et al. (2014). "Do the rich get richer? An empirical analysis of the Bitcoin transaction network". In: *PloS one* 9.2, e86197.

Kroll, Joshua A, Ian C Davey, and Edward W Felten (2013). "The economics of Bitcoin mining, or Bitcoin in the presence of adversaries". In: *Proceedings of WEIS*. Vol. 2013, p. 11.

Meiklejohn, Sarah et al. (2013). "A fistful of bitcoins: characterizing payments among men with no names". In: *Proceedings of the 2013 conference on Internet measurement conference*. ACM, pp. 127–140.

Nakamoto, Satoshi (2008). *Bitcoin: A peer-to-peer electronic cash system*. URL: http://www.bitcoin.org/bitcoin.pdf (visited on 08/20/2018).

Oishi, Shigehiro, Selin Kesebir, and Ed Diener (2011). "Income Inequality and Happiness". In: *Psychological Science* 22.9. PMID: 21841151, pp. 1095–1100. DOI: 10.1177/0956797611417262. eprint: https://doi.org/10.1177/0956797611417262. URL: https://doi.org/10.1177/0956797611417262.

Reid, Fergal and Martin Harrigan (2011). "An analysis of anonymity in the bitcoin system". In: *Privacy, Security, Risk and Trust (PASSAT) and 2011 IEEE Third Inernational Conference on Social Computing (SocialCom), 2011 IEEE Third International Conference on*. IEEE, pp. 1318–1326.

slushpool.com (2018). *Pool payouts*. URL: https://slushpool.com/help/manual/payouts (visited on 10/28/2018).

Stevens, Marc (2012). "Single-block collision attack on MD5." In: *IACR Cryptology ePrint Archive* 2012, p. 40.

Taylor, Michael Bedford (2013). "Bitcoin and the age of bespoke silicon". In: *Proceedings of the 2013 International Conference on Compilers, Architectures and Synthesis for Embedded Systems*. IEEE Press, p. 16.

viabtc.com (2018). *What is auto withdrawal? How can I set it?* URL: `https://support.viabtc.com/hc/en-us/articles/360015954552-What-is-auto-withdrawal-How-can-I-set-it-` (visited on 10/28/2018).

wider.unu.edu (2018). *WIID – World Income Inequality Database*. URL: `https://www.wider.unu.edu/project/wiid-world-income-inequality-database` (visited on 12/14/2018).

wikipedia.org (2018). *Cryptographic hash function*. URL: `https://en.wikipedia.org/wiki/Cryptographic_hash_function` (visited on 10/22/2018).

Wilkinson, Richard (2011). *The spirit level : why greater equality makes societies stronger*. eng. Paperback ed. New York: Bloomsbury. ISBN: 978-1-60819-341-7.

Statutory Declaration

I hereby declare that the thesis with title

Mining concentration and capital accumulation in Bitcoin

has been composed by myself autonomously and that no means other than those declared were used. In every single case, I have marked parts that were taken out of published or unpublished work, either verbatim or in a paraphrased manner, as such through a quotation. This thesis has not been handed in or published before in the same or similar form.

.....	.....
Location, Date	Signature