Open University of the Netherlands
faculty of Management, Science & Technology
Bachelor Computer Science

File dating based on the Physical Location of the File

Author:
Ir. Yvonne Vollebregt
837537914

Researcher:
Ir. Vincent van der Meer
Supervisor:
Dr. Ir. Hugo Jonker

June 28, 2019

Abstract

If files are deleted from a computer then the content of the file is not removed and the file might be reconstructed. In case of reconstructed deleted files the metadata, including the creation and deletion time, is not always available. Information on the creation and sometimes deletion of the file is required to prove the possession of a file. This research delivers methods to derive a creation and deletion time for a file, based on the physical location of the file, which is determined by the blocks allocated. The research is based on the Wildfrag database, that contains the metadata of the files of 220 Windows systems.

If, on a disk, the first, low numbered blocks are written before the blocks beyond, then a timeline with consecutive files having increasing creation times can be put together, and a creation time can be derived from such a timeline. If only part of the files is taken into account to create such a timeline then this is called a partial timeline. Several factors complicate these timelines. In practice, the blocks with the lowest numbers are not always allocated before blocks with higher numbers. Secondly, several file operations disturb a timeline with consecutive files having increasing creation times, because they change the creation time or the allocated blocks of a file. Third, after the deletion of files, holes arise between existing files, and these holes are filled with new files. The effect of the file operations can be excluded by only selecting the files that are a result of the "Create" or "Copy" file operations. In this research it appeared not possible to distinguish the files that are placed consecutively and the files that are placed between existing files.

New methods to derive a creation time based on partial timelines are compared with the existing 10 neighbours strategy. As it is not possible to recognise files that are placed between existing files, the 10 neighbours strategy proves to be the best method to derive the creation time. This method uses the 5 files with preceding blocks allocated, and the 5 files with next blocks allocated, and with these files the mean creation time is calculated. The method results in big differences between volumes, concerning the deviations between the derived and actual creation time. To use the method for a specific volume, first the statistical distribution of these deviations for that volume needs to be determined. Next the certainty is 97,5% that the file is created before the derived creation time plus 2 times the calculated standard deviation. It is possible to determine this for all volumes, but it depends on the size of the standard deviation if the results are practically usable.

The approach to derive a deletion time aims to determine the latest time a file has not been deleted. It is completely based on the best fit allocation strategy, that is applied in Windows 10. If a file consists of as many or less blocks than a deleted file, and uses more space, then that file is created before the deletion of the deleted file. This method is workable, but it is unlikely that the method can only be based on the best fit strategy, and it is not possible to validate the results. Additional experiments with deleted files are required, to further develop the method to determine the latest time a file has not been deleted.

Contents

1 Introduction **4**

2 Background **6**
- 2.1 Metadata . . . 6
- 2.2 Allocation of Files . . . 6
- 2.3 Master File Table . . . 7
- 2.4 Research Database . . . 7
- 2.5 Special file types . . . 8

3 Related Work **10**
- 3.1 Allocation of Files in Practice . . . 10
- 3.2 Modification of Existing Files and Fragmentation . . . 10
- 3.3 Timestamps and File Operations . . . 11
- 3.4 Effect File Operation on Timeline . . . 12
- 3.5 Other Factors that have an Impact on the Timeline . . . 12
- 3.6 Reliability of Timestamps . . . 12
- 3.7 Files and Timestamps in Practice . . . 13
- 3.8 File Dating based on Physical Location . . . 13

4 Methodology **14**
- 4.1 Creation of a Research Dataset . . . 14
- 4.2 Timelines for File Dating . . . 16
  - 4.2.1 Distribution of Files on Volumes . . . 16
  - 4.2.2 Timeline based on Location Files . . . 16
  - 4.2.3 Timeline based on MFT Entries . . . 17
  - 4.2.4 Fit Timeline and Filling Holes . . . 17
  - 4.2.5 Partial Timelines . . . 18
- 4.3 Derivation Creation Time . . . 18
  - 4.3.1 Derive a Creation Time with a Timeline based on Location . . . 18
  - 4.3.2 Derive a Creation Time with 10 Neighbours Strategy . . . 20
  - 4.3.3 Derivation of a Creation Time in the Wildfrag Database . . . 20
- 4.4 Derivation Deletion Time . . . 21
- 4.5 Data Analysis with R . . . 22

5 Results **23**
- 5.1 Characteristics Research Dataset . . . 23
  - 5.1.1 Systems . . . 23
  - 5.1.2 Disks . . . 23
  - 5.1.3 Volumes . . . 24
  - 5.1.4 Files . . . 25
- 5.2 Distribution of Files on Volumes . . . 26
- 5.3 Timeline based on MFT Entries . . . 27

5.4 Fit Timeline and Filling Holes 29
5.5 Partial Timelines 30
5.6 Derivation Creation Time 31
5.7 Derivation Deletion Time 33
5.8 Analysis of the Results 34
5.8.1 Relation Allocation of Blocks and Creation Time of a File 34
5.8.2 Derived Creation Time 34
5.8.3 Derived Deletion Time 35

6 Discussion **36**
6.1 Timelines based on MFT Entries 36
6.2 Factors that affect the degree of deviation per volume 36
6.3 Validation of Results 37
6.4 Reflection on Data Analysis with R 37

7 Conclusions **39**

8 Future Work **40**
8.1 Improvement of Methods Creation and Deletion Time 40
8.2 Related Issues 40

A Description Variables Wildfrag Database **43**

B Validation checks **45**

C Location of files per volume **46**

D Reflection on Process **47**

Chapter 1

Introduction

Files are used as evidence in digital forensic cases. The possession of illegal files, files that are against the law, is punishable. A suspect can anticipate on being accused of the possession of illegal files by deleting possible incriminating files from his computer. However, the content of the file is often not removed, even when the file is deleted. The blocks on the disk where a file was stored are no longer allocated but, as long as the blocks are not overwritten, the file can be reconstructed. Reconstruction of deleted files is done by forensic researchers to gather evidence. In case of reconstructed deleted files the metadata, including the creation and deletion time, is not always available. Therefore it is not clear in which time period the suspect had access to the file. The existence of a file on a disk on its own is insufficient as evidence, as it is possible that a user is not aware of illegal files on his computer. To use files as evidence, the fact that the suspect had access to the file needs to be proven. To do so, fact finders must get to know:

1. Since when the file was on the computer
2. For what time period it was on the computer

The fact that a file has been on a computer since a certain time is important and might be sufficient to build up evidence. The time period adds extra information. For example, if a suspect brings up the objection that he was not aware of a file on his computer, and has deleted the file as soon as possible, then also the time period is relevant.

This research focuses on the location of a file as a means of file dating. The location is determined by the blocks on a disk that are allocated for the file. A file with an unknown creation time, is surrounded with files with known creation times. The question to be answered in this research is: to what extent can files with blocks allocated close to the blocks of a file, be used to derive the creation time of the file? For example, in Table 1.1, is it possible to find $t_?$, given that File 1 is created on t_0 and File 3 is created on t_n?

Table 1.1: File creation times of consecutive files

File 1	File 2	File 3
t_0	$t_?$	t_n

If, on a disk, the first, low numbered blocks are written before the blocks beyond, then a timeline with consecutive files having increasing creation times exists, see Figure 1.1, timeline A. A creation time can be derived from such a timeline. It is clear that File 2 was created before or at t_n.

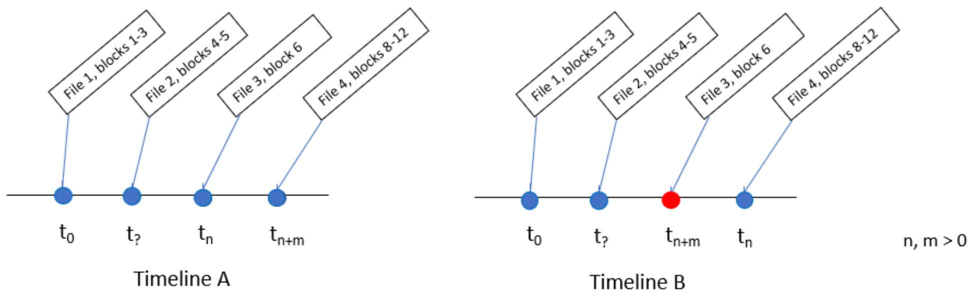


Figure 1.1: Examples timelines

After deletion of files, holes arise between consecutive files, and new files are written between existing files to fill the holes. Especially when disks are actively used, over time a lot of files do not fit in this timeline any longer. For example, File 3 in timeline B in Figure 1.1 does not fit in the timeline. However, in this research timelines are put together to derive a creation time for a file. Besides that, an existing method is analysed. This is a more statistical approach concerning 10 neighbouring files.

To determine the time period a file was on a computer, a deletion time is required next to a creation time. If no metadata is available, then there is no information on the time of deletion of a file. The fact that the file is not yet overwritten, in relation to newly created files, is used to derive the latest time the file has not been deleted.

This research aims to deliver methods to derive a time window consisting of a begin and end time a file was on a disk. If possible the following time window is derived:

(lower bound on file creation, upper bound on file existence)

The research is based on the the research database created by Vincent van der Meer, the Wildfrag database. It contains the metadata of the files of 220 systems, mainly with operating system Windows 10 and file system NTFS. Other operating systems and file systems are not represented sufficiently to draw conclusions on those systems. Therefore the research is restricted to Windows 10 and NTFS. Data analysis is used to analyse the possibilities to derive a creation and deletion time of a file.

The main focus of this research is to provide an overview of all relevant information in the Wildfrag database to create a time window for a file. Visualisations are used to explain the results. The implementation of the results in daily practice and the creation of the accompanying tailor-made visualisations are out of the scope of this project.

The contributions of this research are:

- Development of a method to derive a creation time, based on the physical location of the file, and comparison with the existing 10 neighbours strategy.
- Reliability of the preferred method to derive a creation time, based on performed analyses with the Wildfrag database.
- Development of a method to derive the latest time a file was not deleted.

Chapter 2

Background

2.1 Metadata

Metadata is data that contains information on other data. The file is the main entity in this research. The metadata that is relevant for files in this research consists of:

time values and

location data.

Time values are also called timestamps. Raghavan defines a timestamp as: "the record of the time, according to some reference clock, associated an event" [1]. Each file has several timestamps, which are described in Section 2.3.

The location data of a file is determined by the blocks on a disk that are in use by the file. Blocks have a size that is nowadays usually 4096 bytes per block. Disk space for a file is always allocated in blocks.

In case of hard disk drives, the blocks that are registered in the metadata and the physically allocated blocks are the same. Solid state drives implement wear leveling. This is an algorithm that remaps the block addresses to other physical block addresses in order to extend the life of a storage medium. After application of wear leveling, the physically allocated blocks and the allocated blocks as registered in the metadata are different. This research focuses on the allocated blocks as registered in the metadata, which is called the location of the file. In case of solid state drives this is not the actual physical location of the file, but that is not relevant for this research.

2.2 Allocation of Files

The allocation of files is performed by the file system, as part of the operating system. In this research the file system is NTFS. If the NTFS file system mechanisms would be transparent and clear, then it should be possible to derive time values by reverse engineering.

Unfortunately there is no published specification from Microsoft that describes the NTFS file system, so the exact NTFS allocation strategies are not clear. According to Carrier in 2003 only high-level descriptions of the file system components have been published [2]. Carrier and the Linux NTFS project have published in 2004 and 2005 what they think the on-disk structures are [2]. The official NTFS documentation published by Microsoft is available the following location:

`https://docs.microsoft.com/en-us/previous-versions/windows/it-pro/windows-server-2003/cc781134(v=ws.10).`

It does not contain the exact allocation strategies of files.

Silberschatz describes three strategies to find space for the allocation of a new file [3].

- **First fit**: The first hole that is big enough will be allocated. The system will start either at the first free hole or at the last known first-fit position.

- **Best fit**: The smallest hole that is big enough will be allocated. The entire list must be searched. This will result in the smallest leftover hole.

- **Worst fit**: The largest hole will be allocated. Also the entire list must be searched. This strategy produces the largest leftover hole.

2.3 Master File Table

The metadata of all files and directories is stored in the Master File Table (MFT) [2]. Every file has at least one entry in the MFT. Each entry has an unique MFT entry identification number. After deletion of a file, the MFT entry is reused. Then the sequence number of the MFT entry is increased.

All metadata in the MFT is stored in one of the MFT-attributes. A typical file has the following attributes [2]:

$STANDARD_INFORMATION

$FILE_NAME

$DATA

The $STANDARD_INFORMATION attribute contains, among other things, the primary set of 4 time stamps [2]. These times are the times that are shown after selection of the file properties in Windows. They are 64-bit values and they represent the number of one-hundred nanoseconds since January 1, 1601 UTC [2]. They are described by Carrier as follows [2]:
Creation Time: The time the file was created.
Modified Time: The time the content of the $DATA attribute was last modified.
MFT Modified Time: The metadata last modified time.
Accessed Time: The time the file content was last accessed.
The operating system performs the updating of these time stamps.

The $STANDARD_INFORMATION attribute also contains markers if the file is compressed, sparse, or encrypted [2].

The $FILE_NAME attribute contains the file name and file reference for the parent directory [2]. This $FILE_NAME attribute also contains 4 time values, like the time values as they are described for the $STANDARD_INFORMATION attribute. Windows does not typically update this set of time values like it does with those in the $STANDARD_INFORMATION attribute. According to Carrier they frequently correspond to when the file was created, moved, or renamed [2].

The $DATA attribute is used to store any form of data [2]. It has no format or defined values. As long as a file does not contain too much data, all data is stored within the $DATA attribute in the MFT entry of the file. If the data does not fit any longer in the $DATA attribute then blocks are allocated on the disk to store the data.

2.4 Research Database

The Wildfrag database contains fragmentation data for 'systems in the wild'. To create the research database, the MFT of each system is read. The data is collected with the forensic tool Fiwalk, part of the Sleuthkit and developed by Simson Garfinkel. To collect the data, Fiwalk was controlled by a Python script, running on an USB-drive, with a live Linux distribution. In this way no operating system was active on the researched system, so the measurement was not troubled. Also it was not required to install or delete any software. Fiwalk was configured to read only the MFT (see Section 2.3) and no file content. After the data collection the Python script removed the filenames and filtered the extensions with whitelisting[1] for privacy purposes.

For the data collection only computers with the Windows operating system were taken into account because it was not possible to make the tooling suitable for other devices or other operating systems. The

[1] `https://en.wikipedia.org/wiki/List_of_filename_extensions`

computers of 220 volunteers were used. All volunteers were students at the Zuyd Hogeschool in Heerlen. In general they had pretty new computers that were from 0 to 4 years old.

The inclusion criteria for participation in the research were:

- A laptop or desktop that is currently in use (so no old pc's that are not used any longer)
- A system with running Windows on it (possibly dual boot with another operating system)

The database contains meta-information in five normalised tables:

1. Systems; contains one observation per system (laptop, desktop)
2. StorageDevices; contains one observation per disk. Each system can have more than one disk.
3. Volumes; contains one observation per volume. Each disk can have more than one volumes.
4. Files; contains one observation per file. Each volume will have at least one file but probably much more files.
5. Blocks; contains one observation per file fragment. Non-fragmented files with blocks allocated consist of one fragment, and fragmented files consist of more fragments. Only the files with blocks allocated are included in this table.

The data collection for the Wildfrag database has taken place from October 10, 2018 till January 11, 2019. The time needed for the data collection for one system varied from 0.02 minutes to 16.8 minutes.

The data collection was divided in two runs, 115 during the first run and 105 during the second run. The first run was in October and the second run in the months November, December and January, see Figure 2.1. During the second run the data collection was more extended than during the first run. The operating systems included in the Wildfrag database are Windows 10 and Windows 7, see Figure 2.2. As Windows 10 is the main operating system, the research is restricted to systems having operating system Windows 10.

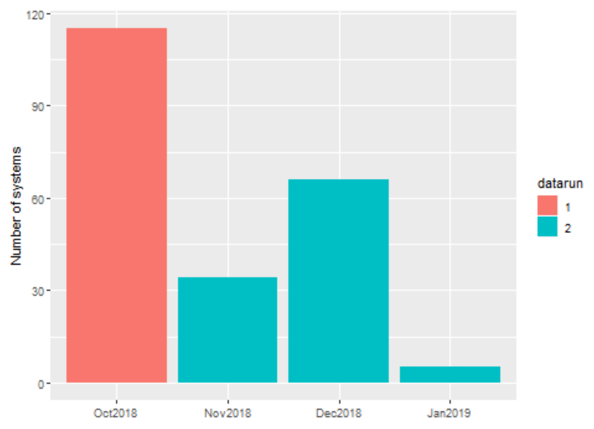


Figure 2.1: Data collection per month (n=220).

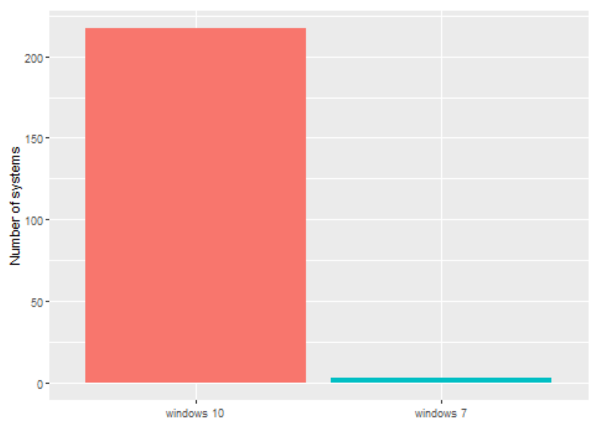


Figure 2.2: Operating systems (n=220).

Appendix A contains a description of the available variables in the Wildfrag database. This description is restricted to the variables that are used in this research.

2.5 Special file types

The research database contains, besides normal files, also special file types. Here the available special file types and the relevance for the research are explained.

Resident files are files that include all data in the $DATA attribute within the MFT table. No blocks are allocated. This file type is not relevant for this research. As soon as a MFT record is deleted then no data and consequently no resident file is left.

Sparse files are files that are partially empty, and attempt to optimise system space by writing information on the empty blocks instead of actually allocating the empty blocks. No or fewer blocks are used, compared with the blocks needed to allocate also the empty blocks. Just the allocated blocks are relevant and are included in the research.

In NTFS several types of **file links** are supported[2]. A **hard link** is a reference to an existing file. Two hard links can have other directory entries and other filenames, but they refer to the same file on disk. Hard links are relevant for the research because blocks are allocated. As different hard links refer to the same blocks, only one of the hard links, the file with the earliest creation time, is used within the research.

A **symbolic link** is a reference to a file system object in the form of an absolute or relative path. The difference between a hard link and a symbolic link is that a hard link refers to a file and a symbolic link contains a reference to another file. For symbolic links no blocks are allocated so the symbolic links are not relevant for this research.

Compressed files. Windows supports the compression of files, folders and volumes. In NTFS, a compressed file is decompressed when it is opened. When the file is closed, it is compressed again[3]. Initially the system reserves disk space for the uncompressed size. After compression the unused space is released[3]. The encryption only takes place on the contents of files. The MFT is not encrypted. Fewer blocks are needed compared with the blocks needed when the data would not be compressed. Just the blocks that are used are relevant and are included in the research.

[2]`https://docs.microsoft.com/en-us/windows/desktop/fileio/hard-links-and-junction`

[3]`http://ntfs.com/ntfs-compressed.htm`

Chapter 3

Related Work

3.1 Allocation of Files in Practice

According to Silberschatz simulations have shown that both first fit and best fit are better than worst fit in terms of decreasing time and storage utilisation. Neither first fit nor best fit is clearly better than the other in terms of storage utilisation, but first fit is generally faster [3].

Limited information is found on the allocation of files in the NTFS file system in practice. Carrier has observed that Windows XP uses the best-fit strategy. The data are placed in a location that will most efficiently use the available space, even if it is not the first available space [2]. Also Bahjat describes in 2019 that the allocation algorithm in NTFS is the best fit allocation algorithm [4].

Sears describes that NTFS attempts to satisfy a new space allocation from the outer band. If that fails, large extents within the free space cache are used. If that fails, the file is fragmented [5].

Casey describes that allocation patterns in actively used NTFS systems are not predictable because the system makes efficient use of the storage space [6].

3.2 Modification of Existing Files and Fragmentation

During the lifecycle of a file, a file might be modified one or more times. As a result the file size can increase or decrease. In case of increase, it is possible that the file does not fit any longer in the allocated space.

Pal describes that different file systems have different strategies to deal with appending and editing files. Some file systems attempt to move the whole file. Other file systems attempt to pre-allocate longer chunks in anticipation of appending. Another technique is delayed allocation. Here the physical allocation of the clusters is delayed until the operating system forces a flushing of the contents [7].

In relation to fragmentation Pal describes that if a file is saved on a disk, and the additional files are also saved starting after the block the original file ended at, fragmentation may occur if the original file is then appended to [7]. This type of fragmentation is called data fragmentation. The file is not written on one set of consecutive blocks but consists of two or more sets of blocks, which are called fragments.

Garfinkel describes that operating systems try to write files without data fragmentation because these files are faster to write and to read. But there are conditions for writing a file with more than one fragment. One of the conditions is: if data are appended to an existing file, there may not be sufficient unallocated blocks at the end of the file to accommodate the new data. In this case some file systems may relocate the original file, but most will simply write the appended data to another location [8].

Casey describes that for a new file fragment not always blocks are allocated after the earlier fragments. NTFS uses the storage space efficiently and sometimes allocates blocks for later fragments in the lower numbered blocks [6].

On the modification of existing objects Sears writes that file systems are optimised for appending or truncating a file. When data are inserted or deleted in the middle of a file, such as when an element is added to an XML document, all contents after the modification must be completely rewritten [5].

The deletion of existing files can lead to external fragmentation. External fragmentation exists when there is enough free space but it is divided in a lot of small holes [3]. Both the first fit and the best fit allocation strategies suffer from external fragmentation [3]. These strategies do not totally fill the free space that exists as a result of the deletion of a file.

3.3 Timestamps and File Operations

The timestamps of a file are changed by operations on the file. Several researchers have performed research on the patterns of timestamps as a result of file operations in the NTFS file system [9, 10, 11, 12, 13, 14, 15].

Cho distinguishes 10 categories of file operations that generate different patterns of timestamps [13] in NTFS. He states that these 10 patterns are a result of regular file operations, and other patterns of timestamps might be a result of timestamp forgery [13].

For this research, especially the file operations that affect the timeline with consecutive files having increasing creation times are relevant. If the creation time of a file or the allocated blocks are changed by a file operation then the file might not fit in the timeline any longer. To recognise these file operations, also the modification time, MFT modified time and accessed time as explained in Section 2.3 are required.

Table 3.1 contains 5 categories of file operations, as distinguished by Cho. From the other 5 categories the 3 file operations "File name change", "File attribute change" and "Move within a Volume" are not relevant because they do not change the location of the file, the creation time and the modification time. The categories "Content Update of General File"and "Content Update of MS Word/Office File" are combined in the category "Content Update" in the Table below. Also the categories "Overwrite with Source File Left" and "Overwrite with Source File Deleted" are combined, into the category "Overwrite". For this research it is not important to distinguish the update of general files and Word files, and to distinguish the overwrite with of without source file left. The effects on the location of the file and on the creation time, in the context of this research, are the same.

In the Table below the following abbreviations are used: crtime for creation time, mtime for modification time, ctime for MFT modified time and atime for accessed time.

Table 3.1: File operations and relevance for ordering of files

File operation, taken from [14]	Relation crtime and mtime	Explanation
"Creation"	crtime = mtime	Crtime, mtime, ctime and atime are equal.
"Copy"	crtime>mtime	Crtime, ctime and atime are changed to the time of operation. Mtime is not changed but originates from the original file.
Content Update	mtime>crtime	Mtime and ctime are changed to the time of operation. Crtime remains unchanged. In case of update of a general file atime remains unchanged, in case of update of a MS Word/Office File atime is updated to the time of operation.
"Move to an Outside Volume"	Existing relation between crtime and mtime remains, can be: crtime<mtime, crtime = mtime or crtime>mtime	Crtime and mtime originate from the original file. Ctime and atime are set to the time of operation.
Overwrite	crtime<mtime or crtime = mtime or crtime>mtime	In case of "source file left" the crtime of the overwritten file does not change. In case of "source file deleted" the crtime of the overwritten file originates from the crtime from the source file that overwrites. Mtime of the overwritten file is changed to mtime of the source file. Ctime is changed to the time of operation and atime remains unchanged.

3.4 Effect File Operation on Timeline

In case of the **creation of a file**, new space is allocated on a disk. If the new space is allocated after all existing files, then the creation and modification time will fit in a timeline with consecutive files having increasing creation times. The timestamp pattern for this operation is distinctive so, with this timestamp pattern, it is clear that only this file operation can have taken place.

In case of **copy of a file**, new space is allocated on a disk. The creation time of the copied file is set to the time of operation. If the new space is allocated after all existing files then the creation time will fit in a timeline with consecutive files having increasing creation times. The timestamp pattern for this file operation is also distinctive.

In case of a **contents update**, the allocation of the blocks might change. The 3 scenarios are:

1. before and after the file update the same blocks are allocated,
2. new blocks are allocated for the whole file and
3. only new blocks are allocated for the new fragment of the file.

If the first scenario is applicable then the file still fits in a timeline with consecutive files having increasing creation times. After scenario 2, the file does not fit in such a timeline. After scenario 3, only the old blocks fit in such a timeline. In conclusion, files that have been changed only partially fit in a timeline with consecutive files having increasing creation times. This type of files does have a distinctive timestamp pattern. As these files can be recognised it is possible to exclude this type of files.

In case of **move from another volume**, new space is allocated on the disk, but the creation time of the file is not changed to the time of operation. After this file operation files do not fit at all in a timeline with consecutive files having increasing creation times. This file operation has a unique timestamp pattern, if also the $FN timestamps (see Section 2.3) are taken into account. Without the $FN timestamps it is not possible to recognise this operation unambiguously. The $FN timestamps are not included in the research database, so it is not possible to exclude this type of files in this research.

In case of the file operation **"Overwrite"** it is assumed that new space is allocated on the disk to write the file, for both scenarios where the source file is left and where the source file is deleted. Probably, in case of the source file left, the blocks of the source file remain allocated and in case of the source file deleted the blocks of the source file are released. In both cases new space is allocated, and the creation time is not set to the time of operation, so these files do not fit in a timeline with consecutive files having increasing creation times. It is not possible to recognise this file operation unambiguously so it is not possible to exclude this type of files.

3.5 Other Factors that have an Impact on the Timeline

According to Bahjat, hard drive defragmentation has a negative effect on the timeline with consecutive files having increasing creation times because during the defragmentation process the "Move" file operation is performed [4]. As mentioned in Table 3.1, in case of the "Move" operation, new blocks are allocated for the file, but the creation time is not updated to the time of operation. Note that if a creation time is based on the allocated blocks, then, in case of a defragmented file, a creation time later in time than the actual creation time of the file is derived. This results in an underestimation of the possession of the file.

Another factor that impacts the timeline is file tunneling, that occurs in Windows 10 [6]. In case of file tunneling the metadata of a recently deleted file is reused for a new file with the same name [6]. Casey describes that file tunneling occurs when the 'Save As' menu item in Microsoft Word is used [6]. Probably the effect of file tunneling on the timeline with consecutive files having increasing creation times is comparable with the disturbing effect of some file operations.

3.6 Reliability of Timestamps

As mentioned before, timestamps are the key values for the derivation of a time window. It is important that they are reliable, in the context of this research.

Douceur describes that system calls enable application programs to set the file creations, modifications and access timestamps to arbitrary values [16]. Agrawal describes that timestamps can be modified by application programs [17]. It is unclear which application programs modify timestamps to arbitrary values.

Knutson describes that the access time cannot be trusted [15]. The updating of the access time can be disabled by the registry. In Windows 7 it was disabled for performance reasons. Also the access time can be changed by other programs like anti-virus software [15]. According to Bahjat in 2019 the updating of the access time is enabled by default in Windows [4].

The MFT modified time should always be at least equal or later than the creation time. In the Wildfrag database, in a lot of cases, the MFT modified time is earlier than the creation time. This is noticed by Jelle Bouma in his research on the metadata of files. Probably the MFT modified time was not updated in a lot of cases and this does not seem to be correct.

3.7 Files and Timestamps in Practice

As this research is performed with systems 'in the wild' it is important to know what is found in practice before. Only few researches are found which are performed on a big amount of 'living' systems in practice. They are described here.

Douceur performed a large-scale study of file-system contents [16]. Agrawal performed a five-year study of filesystem metadata [17]. Conclusions of both researches are:

- On average, file systems are half full. Also over the course of 5 years this only has changed a little.
- File sizes are fairly consistent across file systems, but file lifetimes vary widely and are significantly affected by the user.
- Larger files tend to be composed of blocks sized in powers of two.
- File name extensions are strongly correlated with file sizes, and extension popularity varies with the users.

Meyer described that a large portion of files is rarely modified [18]. In his study, data from 857 desktop computers was collected. A large class of files was written only once (perhaps at install time). Most modified files were modified between one month and a year ago, but about 20% were modified within the last month.

Garfinkel has performed research on file carving on more than 300 hard drives that he acquired on the secondary market [8]. He found that 6% of the files, that was recovered from those hard drives, was fragmented. The percentage of fragmented files per drive was very different. Half of the drives had no fragmented files and 30% of the drives had more than 10% fragmented files [8].

3.8 File Dating based on Physical Location

Bahjat has performed a research on the dating of file fragments [4]. The file fragments in his study are remnants of deleted files, that remain after a new file has replaced the deleted file. He has used the new file to derive an upper bound date for the file fragment [4]. To derive the lower bound date, the creation dates of 10 files with blocks close to the blocks of the file fragment are used [4]. The creation dates are taken from the $FILE_NAME attribute instead of the $STANDARD_INFORMATION attribute, to account for the files that are moved in [4].

This research aims to derive a time window for deleted files that are not overwritten yet. For such an intact deleted file no newer file is available to derive an upper bound date. Also the Wildfrag database does not contain the time values of the $FILE_NAME. Therefore not all Bahjat's methods can be used in this research.

Chapter 4

Methodology

4.1 Creation of a Research Dataset

In this research, the Wildfrag database, as described in section 2.4, is used. This database contains the raw data, which is collected from 220 systems. The purpose of this first research phase is to create a research dataset with only the relevant data from this database, to perform all analyses for this research. The starting points of the creation of the research dataset are:

- All groups or categories in the dataset must be represented sufficiently, to make it possible to draw conclusions on that group or category. For example, the Wildfrag database contains only 3 Windows 7 systems, too few to draw conclusions for this subgroup. Therefore these systems are excluded.
- The data must be relevant for the research. All variables that are not relevant, are not taken into account. For example, the file type is not analysed, so the extension of the file is not included.
- The data must be reliable. Data that does not seem to be correct is not taken into account.

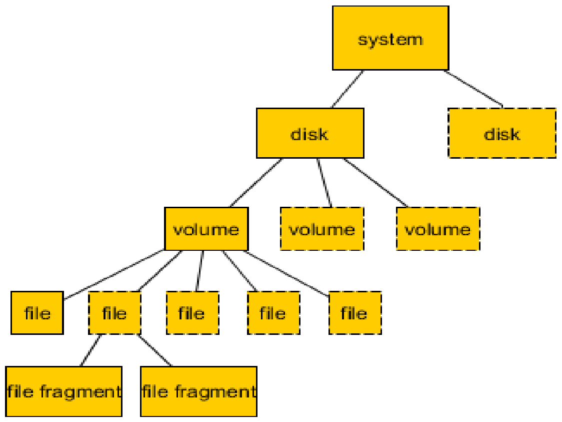


Figure 4.1: Data hierarchy Wildfrag database

The data hierarchy in the Wildfrag database consists of systems, disks, volumes, files and file fragments, see Figure 4.1. Only files with blocks allocated consist of one or more file fragments. If a file consists of more file fragments then that file is fragmented.

A system contains one or more disks. A disk contains one or more volumes and a volume normally contains several files. Figure 4.1 shows the hierarchy of a system with a very limited content, as volumes can contain thousands of files. In this figure the disk, volumes and files with the dashed lines are optional.

For these entities several characteristics are described, see Table 4.1. These characteristics are mainly used to include only the subgroups that are represented sufficiently.

Table 4.1: Overview of characteristics to be described per entity

Entity	Characteristic
Systems	S1: Frequencies operating systems S2: Number of disks S3: Number of volumes S4: Number of files
Disks	SD1: Frequencies disk types SD2: Number of volumes SD3: Number of files
Volumes	V1: Frequencies file systems V2: Frequencies block sizes V3: Number of files V4: Distribution number of files V5: Occupation rate V6: Frequencies occupation rate categories V7: Frequencies disk types
Files	F1: Frequencies special files F2: Frequencies number of hard link copies F3: Frequency fragmented files F4: Characteristics timestamps F5: Frequencies file operations

In case of frequencies (S1, SD1, V1, V2, V6, V7, F1-F3 and F5), the numbers and percentages per group are presented. The number of files (S4, SD3, V3) contains all files, including files without blocks. The occupation rate categories (V6) are: 0-25%, 25-50%, 50-75% and 75-100%. To describe the disk types per volume (V7) primary and secondary hdd's are distinguished. A hdd is a primary hdd if the hdd is the only disk on a system, and it is a secondary hdd if the hdd exists next to a sdd.

In case of numbers (S2-S4, SD2-SD3 and V3), the n (number of observations), minimum, mean and maximum are presented.

To describe the distribution of the number of files (V4), the amount of volumes per category is presented, with categories: $<$10, $\geq$10 and $<$100, $\geq$100 and $<$1.000, $\geq$1.000 and $<$10.000 and $>$10.000 files.

To calculate the occupation rate (V5) only one hard link of each set of hard links is included. The occupation rate is calculated as:

(number of blocks * block size) * 100 / size

The minimum, mean and maximum occupation rate are presented. For the timestamps (F4) are presented: the minimum and maximum values, number of timestamps in the future and number of incomplete timestamps. The file operations that are described (F5) are "Creation", "Copy" and "other file operations". All characteristics specified in Table 4.1 are described in Section 5.1.

The selection of the relevant variables for the research is based on the planned analyses. Only the variables that are required for the analyses are included in the dataset. Appendix A contains an overview of the variables included in the research dataset.

To include only the reliable data, several validation checks are performed. The objectives of the validation checks are:

- verify that the contents of the variables corresponds with what is expected based on the definition
- and exclude observations with values that not seem to be correct.

Checks are programmed like if there is any time stamp in the future, there should be none. The performed checks are described in Appendix B. The exclusions based on the checks are:

1. Systems that contain $\leq$ 250 files.
2. Files that contain $<$10 files.

3. Files with a creation time in the future.
4. Files with a missing or incomplete creation time.

4.2 Timelines for File Dating

4.2.1 Distribution of Files on Volumes

The usage of the location of a file for file dating is based on the idea that neighbouring files also have close creation times. Based on the literature, it is assumed that files are allocated starting with the lower block numbers and that holes, which arise because of the deletion of files, are filled according to the best fit algorithm. If this is true then, depending on the occupation rate, much more blocks with low numbers than blocks with high numbers are allocated.

To check this assumption, the distribution of the allocated blocks is analysed. Per volume, the total amount of blocks is divided in 100 equal parts. The percentage allocated blocks is calculated for each part. Therefore the number of allocated blocks is divided by the total number of blocks per volume part. This is done per occupation rate category, and for 12 individual volumes. The occupation rate categories are: an occupation rate of 0-25%, 25-50%, 50-75% and 75-100%. The individual volumes are chosen as follows: a primary hdd, a secondary hdd and a ssd per occupation rate category. Graphs are created with on the x-axis the volume parts and on the y-axis the percentage of allocated blocks. The graphs per occupation rate category are presented in Section 5.2 and the graphs of the individual volumes in Appendix C.

The graphs do not exactly present the location of the files on a volume because the MFT is missing. The MFT is also a file, but the metadata of the MFT is not included in the Wildfrag database. Furthermore, the graphs just present a snapshot of the situation on the disk. It is assumed that the disk is written starting with the blocks with the low numbers, but if a lot of files are created and deleted afterwards, then the view is more diffuse.

4.2.2 Timeline based on Location Files

As mentioned before, a creation time can be derived from the timeline with consecutive files having increasing creation times, if the lower block numbers on a disk are written before the higher block numbers. Two main complications to use this timeline to derive the creation time are:

1. Several file operations disturb the timeline by changing the allocated blocks or the creation time.
2. Files that are placed between existing files, to fill holes that are a result of the deletion of files.

Besides these complications more factors exist. Among them are at least the factors defragmentation and file tunneling, which are described in Section 3.5. The effect of defragmentation can be traced back to a disturbing file operation and probably this also applies to the effect of file tunneling.

Despite the complications it is possible to put together a timeline with consecutive files having increasing creation times, to derive a creation time. To deal with the complications the following approach is used:

- Only the files that are a result of a file operation that does not disturb the timeline are included in the timeline with consecutive files having increasing creation times. The file operations that do not disturb the timeline (see Section 3.4), are:
    1. Creation: crtime = mtime = ctime = atime
    2. Copy: crtime = ctime = atime and crtime > mtime
- Only the files that are not fragmented are included because fragmented files have allocated more than 1 series of connecting blocks and it is unclear which of these series fits in the timeline.
- Due to at least the files that are placed between existing files, not all resulting files will fit in this timeline. Therefore only the files with a creation time that fits in the timeline are taken into account.

- The files that fit in the timeline are identified by consecutively searching for the earliest time. The file with the earliest time is the first file of the timeline. Files with earlier blocks allocated do not fit. Next the earliest time after the first creation time is looked for. Possible files that have blocks allocated between the earliest and the next file do not fit. And so on.

Attention needs to be given to the programming algorithm to derive the timeline. A possibility is to search for the minimum creation time of each file, starting with the file with the lowest block number. In that case n + (n-1) + (n-2) + ... + 1 comparisons need to be done, so this algorithm takes O(n^2) time. Another option is to start with the last file, with the highest block number. The creation time of the last file is taken as the minimum creation time. For the last-but-one file this minimum up till then is compared with the creation time of the last-but-one file. Each creation time of each next file is compared with the minimum up till then. In this case the algorithm takes only O(n) time.

After the restriction to only 2 file operations, still no clear timeline exists, because of the files that are placed between existing files. The analysis to determine till what extent the files, which are placed between existing files, can be recognised, is described in Section 4.2.4. To verify that the effect of the file placing between existing files is the main disturbing effect on the timeline, after the restriction of file operations, a test with the MFT is performed. As long as the MFT entry is used for the first time, later created files will have a higher MFT entry number. In the next section this analysis is described.

4.2.3 Timeline based on MFT Entries

The analysis described here is supportive to the research. The purpose is to verify that the effect of the first complication as described in Section 4.2.2, is eliminated by only including the files that are a result of the "Create" and "Copy" file operations, also described in Section 4.2.2.

For this analysis, only the MFT entries which are used for the first time are taken into account. They can be recognised by the sequence number. The sequence number is 1 for these entries, because only after reuse of the MFT entry, the sequence number is increased.

For the analysis are included the files with blocks, with a complete creation date that is not in the future. Both files that are fragmented and not are included, because each fragmented file has only 1 MFT entry.

It is assumed that, using the MFT entries, the effect of the files that are placed between existing files is excluded. The hypothesis is: if the MFT entry is used for the first time and only the file operations "Create" and "Copy" are included, increasing MFT entries have increasing creation times. To test this hypothesis, the number of MFT entries that fit in this ordering is counted. The following selections are applied:

1. All files are included.

2. Only files after the file operations "Create" and "Copy" are included.

If the percentage of MFT entries that fits in the ordering of the second data selection is close to 100%, then this indicates that the disturbing effect of the file operations is eliminated by only taking into account files that are a result of the file operations "Create" and "Copy".

The percentages of 12 individual volumes and overall are presented in Section 5.3. Also graphs are created with on the x-axis the MFT entry number and on the y-axis the creation time, for 2 arbitrary volumes. For each volume 2 graphs are created, for all files and for the created and copied files. These graphs show the ratio between files that fit in the timeline and not.

4.2.4 Fit Timeline and Filling Holes

As mentioned in Section 4.2.2, files that are inserted between existing files, disturb the timeline, even if only files after specific file operations are taken into account. The questions to be answered are:

1. Does a substantial amount of files fit in the timeline with consecutive files having increasing creation times, if only the files after the "Create" or "Copy" file operation are taken into account and

2. if so, to what extent are the files that are placed between existing files, recognisable?

Maybe they can be recognised because a hole between the file and the next file exists. Probably, files that are placed consecutively do not have unallocated blocks between them.

Each file is marked if it fits in the timeline or not. Also, each file is marked if a hole exists between the file and the next file. A hole exists if:

”first block number of the next file” - ”last block number of the file” > 1

To answer the questions, per volume and overall the following percentages are calculated:

- Number of files that **fit** in the timeline, **with** holes behind
- Number of files that **fit** in the timeline, **without** holes behind
- Number of files that **do not fit** in the timeline, **with** holes behind
- Number of files that **do not fit** in the timeline, **without** holes behind

4.2.5 Partial Timelines

The procedure to select the files that fit in the timeline, as described in Section 4.2.2, is strict. If, occasionally, a file with an early creation time is placed at the end of the disk, then only a few or no files fit in the timeline. To circumvent this problem, partial timelines are created. A partial timeline is a timeline that is based on 0.5n files before, and 0.5n files after a specific file. The question is what n should be to create meaningful timelines. A small n (<500) probably corresponds with a higher percentage of files that fits in the timeline then a big n (>5000). On the contrary, probably, a small n results in a less precise result when using the timeline to derive a creation time. Therefore in this research experiments are done with n=1000 and n=100.

All included volumes are divided in parts of 1000 and of 100 files. The remaining part per volume, with less than 1000 or 100 files, is not taken into account. Per part, a timeline with consecutive files having increasing creation times is created, as described in 4.2.2. Per part, the number of files that fits in the timeline is counted. The minimum, mean and maximum number of files that fits in the partial timeline, for both n=1000 and n=100, are presented in Section 5.5. For 4 volume parts (2 with n=1000, 2 with n=100) a graph is created with on the x-axis the creation time and on the y-axis the number of the first block of the file. The files that fit and do not fit are distinguished by colour.

4.3 Derivation Creation Time

4.3.1 Derive a Creation Time with a Timeline based on Location

To derive a creation time of a file with a timeline, the blocks allocated for the file are compared with the blocks allocated for the files in the timeline. In this way the previous and next file in the timeline are determined. Next several possibilities exist to determine the creation time based on the previous and next file of the timeline:

1. the creation time of the previous file. In this case the derived time is equal to or **earlier** than the actual time the file was created.
2. the creation time of the next file. In this case the derived time is equal to or **later** than the actual time the file was created.
3. a **combination** of one or more previous and one or more next files. This probably results in the best approximation of the creation time, but it is not absolutely sure that the concerning file existed at the derived creation time.

Table 4.2: Example derivation Creation Time from timeline

No	Start block	Creation time	Fit in timeline
1	2527	2017-03-12 15:52:29	False
2	3506	2016-02-10 19:28:56	False
3	3534	2016-02-10 19:29:38	True
4	3537	2012-03-08 10:21:49	
5	3560	2016-02-10 19:44:13	True
6	4290	**to be derived**	
7	4291	2016-03-09 15:57:01	True
8	4301	2017-01-12 20:19:12	False
9	4379	2016-04-04 21:19:46	True
10	4761	2017-08-04 12:22:04	
11	4782	2016-04-04 22:42:38	True

Table 4.2 visualises the above described approach. The white row contains the concerning file. The blue rows are the files that fit in the timeline. The red rows contain files that do not fit in the timeline. The gray rows contain files that are excluded before the creation of the timeline, because they are not a result of the "Create" or "Copy" file operation.

If, in this example, the derived creation time is based on the second approach, on the next file, then the derived creation time is: 9-3-2016 15:57:01.

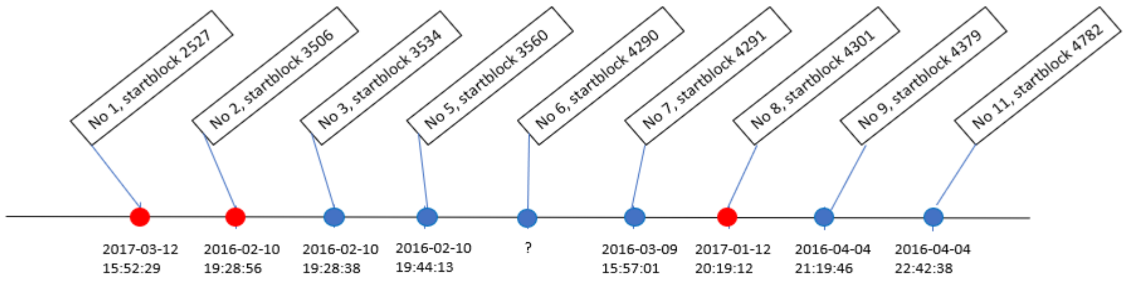


Figure 4.2: Timeline Example Table 4.2

Figure 4.2 shows the files of Table 4.2 in a timeline. Figure 4.3 shows the location versus the creation time of the files. The blue files are again the files that fit in the timeline and the red files the files that do not fit.

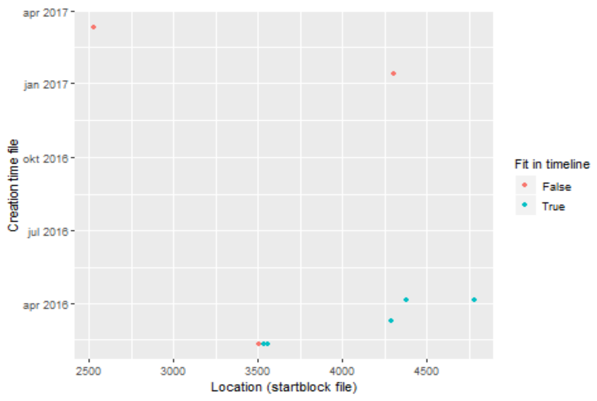


Figure 4.3: Timeline Example Table 4.2

4.3.2 Derive a Creation Time with 10 Neighbours Strategy

Another approach to derive a creation time is to calculate the mean creation time of k neighbours, files that have blocks allocated next to the concerning file. Bahjat has performed this strategy in his research, with the 10 files preceding the concerning file [4].

Table 4.3: Example derivation Creation Time based on 10 neighbours

No	Start block	Creation time	Derived creation time
8	596	20-9-2018 19:58	
9	680	12-9-2018 17:16	
10	684	14-9-2018 10:14	
11	750	20-9-2018 12:10	
12	777	27-8-2018 19:01	
13	778	4-9-2018 10:48	
14	978	11-9-2018 15:42	
15	1464	5-9-2018 12:45	
16	1537	to be derived	11-9-2018 08:33
17	2075	4-9-2018 07:48	
18	2216	30-9-2018 14:44	
19	2362	14-9-2018 08:14	
20	2368	01-2-2019 22:11	
21	2395	21-9-2018 20:50	
22	2747	3-9-2018 11:27	
23	3668	19-10-2018 13:57	
24	3816	07-09-2019 10:56	

Bahjat writes that 10 neighbours are sufficient to calculate the mean value. It is unclear why he has taken only preceding files. To get the best approximation probably 5 preceding and 5 next files should be taken. In this research, in case of 10 neighbours the 5 preceding and 5 next files are taken.

Table 4.3 visualises this approach. Again, the white row contains the concerning file. The blue rows are the 10 neighbouring files, and the red rows the files that are not taken into account. The gray rows contain files that are excluded because they are not a result of the "Create" or "Copy" file operation.

The question is which strategy results in a better derived creation time, the "partial timelines strategy" or the "10 neighbours strategy". The partial timelines strategy only takes files that fit in the timeline into account, so the reference points are better. If the concerning file is placed consecutively, then the partial timelines strategy probably results in a more accurate creation time. On the contrary, if the concerning file is placed between existing files, then this strategy results in a creation time before the actual creation time, because this strategy is based on files that are placed consecutively and the file filled a hole arisen after the deletion of a file. The 10 neighbours strategy is a more statistical approach and is not related to files placed consecutively or files placed between existing files. In this research both strategies are compared.

4.3.3 Derivation of a Creation Time in the Wildfrag Database

To derive a creation time for a file, the "partial timeline strategy" and the "10 neighbours strategy" are compared using the data of the Wildfrag database.

For the partial timeline strategy, partial timelines with n=1000 and n=100 are created, as described in Section 5.5. Only the files that are located after the first file that fits in the timeline are taken into account because only for these files a creation time can be derived based on the timeline. Therefore only the files with a first block number $\geq$ the first block number of the first file in the timeline are included. Then the previous and next file, based on the first block per file, are taken from the timeline. If the file itself fits in the timeline, then the previous file is equal to the concerning file. Otherwise the previous file is a file with a lower first block number.

In this research the approaches 2 and 3, as described in Section 4.3.1, are applied. To calculate the mean value in approach 3, 1 previous and 1 next file are used.

To derive the creation time based on the 10 neighbours strategy, the mean of the creation times of the 5 preceding and 5 next files is calculated. The preceding and next files are the files with the block numbers before and after the file. Per volume the first 5 and last 5 files are excluded because they do not have 5 preceding and 5 next files. For this analysis also only the files that are a result of a "Create" or "Copy" file operation are included, because only these files have a creation time that can be used as reference.

To compare the methods, for all files for which a creation time is derived, the deviation of the derived creation time versus the actual creation time is calculated as:

deviation = (derived creation time - actual creation time), in days

For both methods, and per variant, the overall mean of the deviations of the derived creation time versus the actual creation time is calculated. Also the standard deviation is calculated as a measure of the statistical distribution. The basic principle is that a normal distribution of the deviations may be assumed. Next the mean and standard deviation of the deviations are used to determine a derived creation time with a certainty of 97,5%. Therefore the mean deviation and 2 times the standard deviation are added to the derived creation time.

derived creation time 97,5% certainty = derived creation time + mean deviation + 2 * sd deviation

In case of the 10 neighbours strategy it is expected that the mean deviation is about 0. Then only 2 times the standard deviation of the deviations needs to be added to the derived creation time.

4.4 Derivation Deletion Time

The fact that a deleted file is not overwritten yet, is an indication that several other files are written before. Otherwise the blocks released because of the deletion of the file would be allocated for these files.

As described in Section 2.2, in NTFS the best fit allocation strategy is used. In the example below it is clear that, if best fit allocation is applied, file x is deleted after the creation of file y. Otherwise, if the blocks 11-20 would be available at the time file y was written, then the blocks 11-18 would be allocated.

Table 4.4: Example Deletion of Files and Reallocation of Blocks

files	deleted file x 11-20	files	new file y 61-68	hole 69-75	files

The blocks that are released in case of deletion of a file might be followed by unallocated blocks. For a new file the whole space is available. This total available space is indicated by n;

n: the sum of the number of blocks allocated for the file and the number of unallocated blocks after the file

To determine which files are created before the deletion of a file, 3 groups of files are described here (see Figure 4.4):

1. k (= number of blocks allocated) + (n-k) (= number of unallocated blocks before the next file) = n
2. k = n, followed by m-n unallocated blocks until the next file
3. k<n and n<m

Note that 1 and 2 overlap if n=k and m-n=0. In that case a file exactly allocates the amount of blocks available between 2 existing files. If such a hole exists then the blocks of this hole will be allocated when applying the best fit strategy.

A deleted file is called d. In situation 1, blocks are allocated in exactly the same space as the space of file d. In this situation it is not clear if a new file has allocated blocks before or after the deletion of file d.

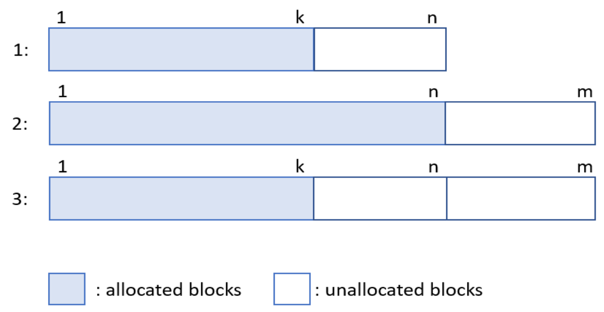


Figure 4.4: Files relevant for derivation deletion time.

It is possible that 2 holes of the same size were available at the same time. In that case it is not clear which hole is filled first, when applying the best fit strategy.

In situation 2, space m is used, with m-n as small as possible. After deletion of file d, space n is available, so files of size n, with space m, and m>n, are created before the deletion time of file d.

In situation 3, space m is used to allocate blocks for a file z with size k. If file d was deleted at the creation time of file z, then not space m but space n would be chosen to allocate blocks. In this situation it is clear that file z is created before the deletion of file d.

To find the files that are created before the deletion of a file, the files as described at 1 are not taken into account because they are not certainly created before the deletion of a file. The files at 2 and 3 are selected together as follows:

m>n and k≤n

All files that meet this criterion are selected, and the maximum creation date is determined. This is the latest time the file has not been deleted.

This method might not work for files that are created and shortly after deleted. Then no new files that meet the criterion are created between the creation and deletion of a file and therefore no reference files are available. In that case only a creation time may be derived. Probably this only happens with recently created files. Then it is only clear that deletion happened between creation and the time of investigation.

In this research the above described method is applied to 4 files, with and without holes behind, and with different sizes. The results for these files are described in Section 5.7.

4.5 Data Analysis with R

To analyse the Wildfrag database, the programming language R is used. R is used by a lot by statisticians and data miners for data analysis and statistics. R is comparable to other commercial statistical packages, such as SAS and SPSS, but R is available at no charge. According to Field, the main advantages of using R are that it is free, and that it is a versatile and dynamic environment. It is a rapidly expanding tool and can respond quickly to new developments in data analysis [19]. Field describes that the disadvantage of R is the ease of use as it works with a command line rather than a graphical interface [19]. Obviously, this is not a problem for programmers.

The Tidyverse is the part of R that is mainly used. According to Wickham that is ”a collection of R packages designed to work together to make data science fast and fluent” [20]. Using the Tidyverse helps to start quickly with the analyses. Besides the Tidyverse also specific functions are created with Base R.

R is used to create the research dataset, to derive new variables required for the analyses, and to perform the analyses. The created R code used to achieve the results is delivered at the end of the project. This makes it possible to repeat the analyses and the code also may be used as a starting point for future analyses by researchers or other students.

The usage of R for this project is evaluated. In Section 6.4 the experienced advantages and disadvantages are described.

Chapter 5

Results

5.1 Characteristics Research Dataset

5.1.1 Systems

The research database contains 3 systems (1%) with the operating system Windows 7, see Figure 2.2. As the research is restricted to Windows 10, these 3 systems are excluded from the research. The characteristics of the remaining systems are presented in Table 5.1.

Table 5.1: Characteristics systems (n=217)

Characteristic	N	Minimum	Mean	Maximum
Number of disks	331	1	1.5	2
Number of volumes	725	1	3.3	7
Number of files	99.663.220	4	459.278	1.742.137

Five systems contain less than 250 files. Possibly this small amount of files is caused by Windows Bitlocker. During the data collection several systems appeared to have Windows Bitlocker and volumes with Windows Bitlocker could not be read. A new Windows 10 installation in Virtual Box consists of more than 60.000 files, so 250 files is by far not enough to contain an operating system. Therefore these 5 systems are excluded from the analyses.

The total amount of systems included is: **212**.

5.1.2 Disks

All systems consist of 1 or 2 disks, which are rotational hard disk drives and solid state drives. The characteristics of these disks are presented in Table 5.2.

Table 5.2: Characteristics storage devices (n=326)

Characteristic	N	Minimum	Mean	Maximum
Number of volumes	715	1	2.2	5
Number of files	99.662.639	6	305.714	1.742.137

The database contains substantial amounts of both solid state drives (ssd) (56%) and hard disk drives (hdd) (44%), see Figure 5.1. All systems contain 1 hdd, 1 ssd, 1 hdd and 1 ssd or 2 ssd disks, see Figure 5.2. Most systems (52%) contain 1 ssd and 1 hdd.

The total amount of disks included is: **326**.

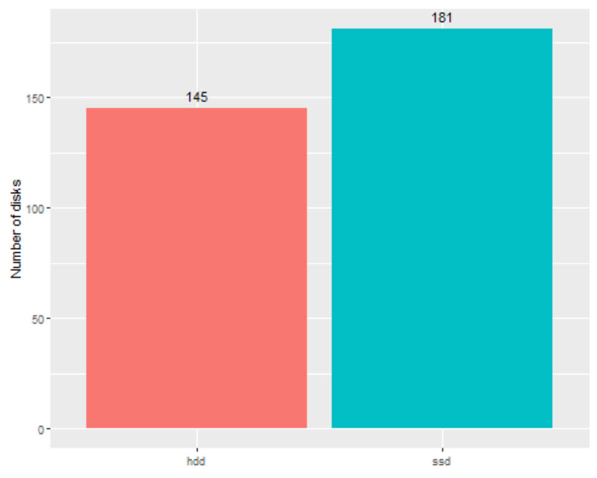


Figure 5.1: Overview of rotational and not rotational disks (n=326).

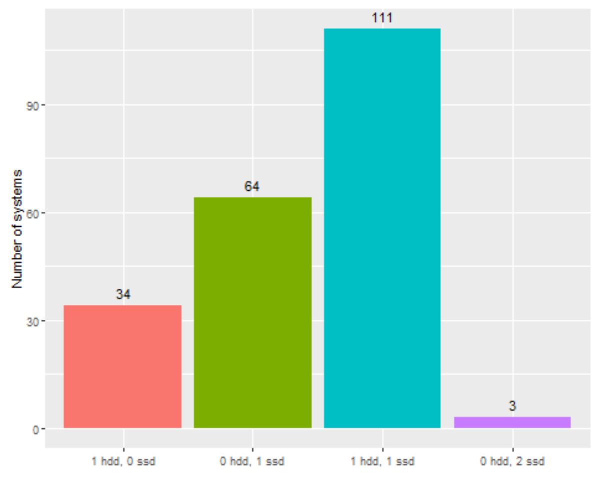


Figure 5.2: Rotational and not rotational disks per system (n=212).

5.1.3 Volumes

The main file system is NTFS, see Figure 5.3. Only 4 volumes (0.6%) have the file system Ext4. The research is restricted to NTFS and these 4 volumes are excluded from all further analyses.

The block sizes of the volumes are presented in Figure 5.4. 97% of the volumes has block size 4096. Therefore the research is restricted to block size 4096. The 22 volumes with other block sizes are excluded from all further analyses. After these exclusions the research dataset contains 689 volumes.

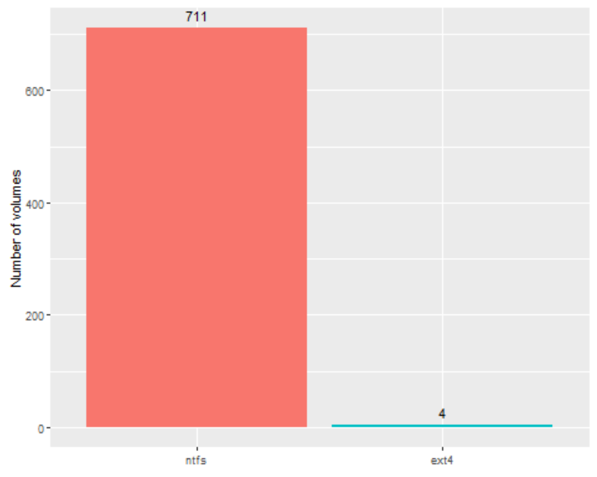


Figure 5.3: Overview of file systems (n=715).

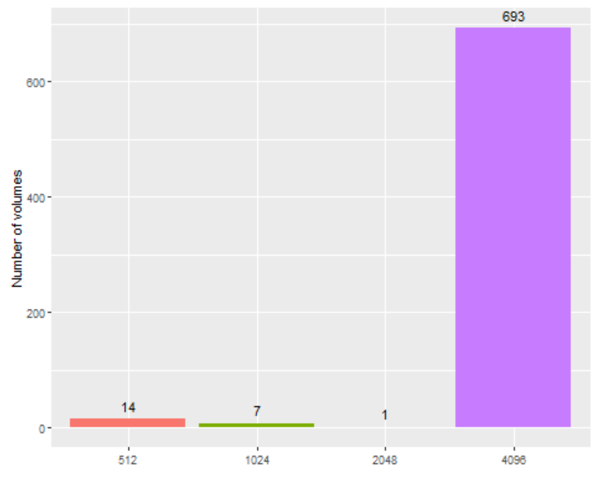


Figure 5.4: Overview of block sizes (n=715).

The characteristics of the volumes are presented in Table 5.3. The occupation rate is an approximated value because the size of the MFT is not included in the calculation. Therefore the actual occupation rate is higher than the presented value.

Table 5.3: Characteristics volumes (n=689)

Characteristic	N	Minimum	Mean	Maximum
Number of files	90.995.399	1	132.069	1.624.774
Occupation rate	*not applicable*	0	38.6	90.7

Big differences exist between the number of files per volume, see Figure 5.5. 36% of the volumes contains <10 files. These volumes are excluded from all further analyses.
The total amount of volumes included is: **439** volumes, with 90.994.118 files.

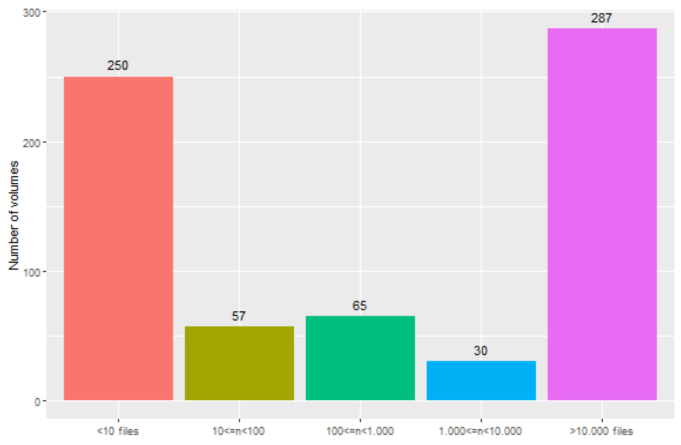


Figure 5.5: Distribution number of files per volume (n=689).

The 2 tables below, Tables 5.4 and 5.5, show the frequencies of the disk types and occupation rates of the volumes.

Table 5.4: Frequencies disk types (n=439)

Disk type	N	Percentage
Primary hdd	59	13%
Secondary hdd	156	35%
Ssd	224	51%

Table 5.5: Frequencies occupation rate categories (n=439)

Occupation rate category	N	Percentage
0-25%	210	48%
25-50%	115	26%
50-75%	68	15.5%
75-100%	46	10.5%

5.1.4 Files

The Wildfrag database contains special file types, as described in Section 2.5. The frequencies of the special file types are presented in Figure 5.6.

The hard links contain several references to the same file. Most of the times, 2 hard link copies with the same hard link identification exist, but also 1 hard link with 962 copies exists, see Figure 5.7. If only 1 hard link copy per file is kept, then only 41% of the hard links remains (7.139.669 out of 17.229.004 files).

The files which are relevant for the research are:

56.280.882 'other' or normal files and

7.139.669 unique hard link files.

In total: 63.420.551 files.

Only these files are taken into account in the research. Part of these files, 2.5%, is fragmented.

As mentioned before, the creation time is a key variable in this research. Several selected files have a creation time that cannot be used:

18.399 (0.0003%) files with an incomplete creation time and

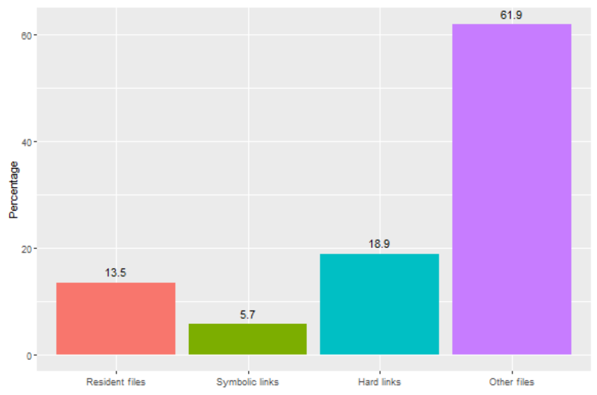


Figure 5.6: Special files (n=90.994.118).

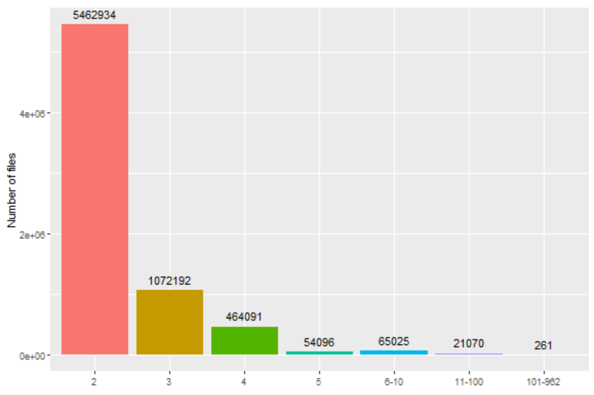


Figure 5.7: Hard link copies (n=17.229.004).

72 files with a creation time in the future.

After exclusion in total: 63.402.080 files.

These files are not taken into account in analyses where the creation time is involved.
Table 5.6 below contains characteristics of files.

Table 5.6: Characteristics files (n=63.402.080)

Characteristic	Minimum	Mean	Maximum
Number of blocks	1	148	488.282
Creation time	1970-01-01 00:00:27	2018-01-18 22:07:58	2019-01-11 12:53:07

For the analyses especially the files with the file operations "Create" and "Copy" are important. These file operations can be distinguished by their timestamp pattern. Table 5.7 contains the frequencies of these file operations, based on the timestamp patterns of the files.

Table 5.7: File operations (n=63.402.080)

File operation	Number of files	Percentage
"Create"	8.039.429	13%
"Copy"	11.575.220	18%
Other file operations	43.787.431	69%

The total amount of files included is: **63.420.551**.
When only files that are a result of the file operations "Create" and "Copy" are included: **19.614.649**.

5.2 Distribution of Files on Volumes

Figures 5.8-5.11 show the mean distribution of the files on a volume, per occupation rate category. The 4 different occupation rate categories (0-25%, 25-50%, 50-75% and 75-100%) are described in Section 4.1. As described in Section 4.2.1, each volume is divided in 100 equal parts and per part the percentage of allocated blocks is calculated. These 100 percentages of allocated files are visualised in a graph. Appendix C contains 12 graphs of individual volumes. The graphs here contain the mean distribution of the files, broken down to the occupation rate category.

The blue planes visualise the distribution of files if the first part of the volume is fully occupied. For example, for an occupation rate of 25% this means that all blocks of the first 25 volume parts are allocated.

Note that the blocks allocated for the MFT are missing in these graphs because the Wildfrag database does not contain the blocks allocated for the MFT.

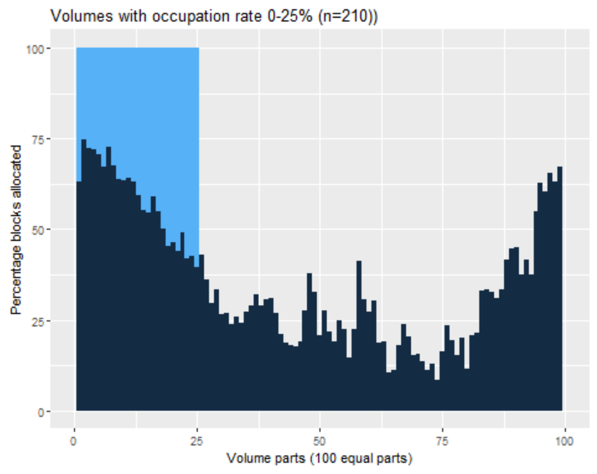


Figure 5.8: Occupation rate 0-25%.

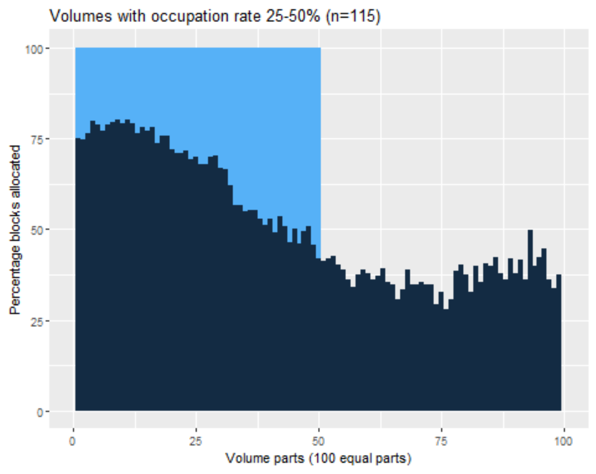


Figure 5.9: Occupation rate 25-50%.

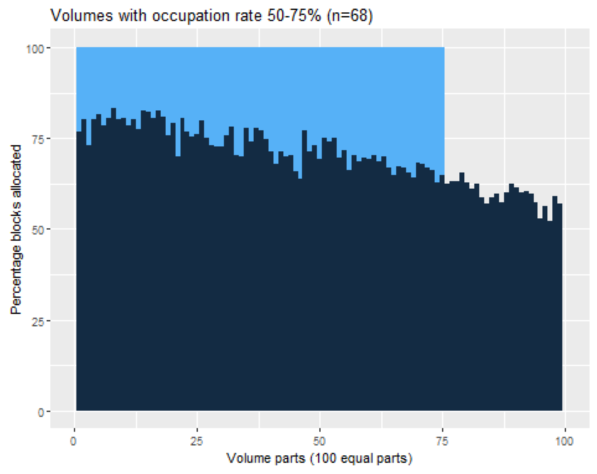


Figure 5.10: Occupation rate 50-75%.

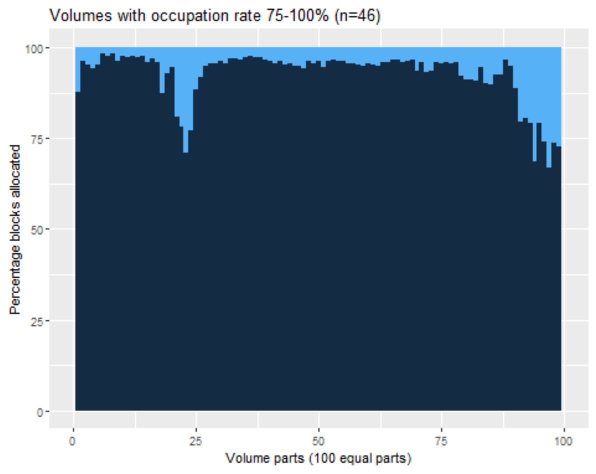


Figure 5.11: Occupation rate 75-100%.

5.3 Timeline based on MFT Entries

For this analysis the numbers of files that fit in the timeline based on MFT entry numbers are counted. The files that fit in this timeline are files with increasing MFT entry numbers, having increasing creation times. The analysis is performed for both all files and for the files that are a result of the "Create" or "Copy" file operation. Table 5.8 contains the results for 12 volumes and Table 5.9 contains the overall results.

Table 5.8: Fit in timeline based on MFT entries per volume, 12 volumes

Volume	All files selected		Only after "Create" or "Copy" file operation	
	No. files fit	No. files no fit	No. files fit	No. files no fit
1	1780 (81%)	426 (19%)		
3	95657 (55%)	78652 (45%)	1 (0%)	50773 (100%)
4	0 (0%)	3 (100%)	0 (0%)	1 (100%)
5	10902 (99%)	104 (1%)	0 (0%)	3091 (100%)
8	67255 (100%)	112 (0%)	1 (0%)	5044 (100%)
9	29 (53%)	26 (47%)	0 (0%)	2 (100%)
10	28529 (100%)	26 (0%)	0 (0%)	5048 (100%)
11	51347 (46%)	59223 (54%)	2 (0%)	20900 (100%)
13	0 (0%)	3158 (100%)	0 (0%)	429 (100%)
14	169 (99%)	1 (1%)	0 (0%)	2 (100%)
15	101538 (89%)	12706 (11%)	2 (0%)	32940 (100%)
18	15907 (94%)	1080 (6%)	1 (0%)	1807 (100%)

Table 5.9: Overall fit in timeline based on MFT entries, 437 volumes

Selection files	Volume		Percentage fit in timeline based on MFT entries				
	N	Mean no. of files	Minimum	Mean	Maximum	Standard deviation	Median
All files	437	31.670	0%	41%	100%	43%	19%
After "Create" or "Copy" file operation	375	10.143	1%	96%	100%	17%	100%

Figures 5.12-5.15 show the files that fit in the timeline based on MFT entry numbers. The blue files are the files with increasing MFT entry numbers having increasing creation times, and form a so called timeline based on MFT entries. The red files are the files which have a creation time later than the creation time of a file with a higher MFT entry number. Figures 5.12 and 5.14 contain all files and figures 5.13 and 5.15 contain only the files that are a result of the "Create" or "Copy" file operation.

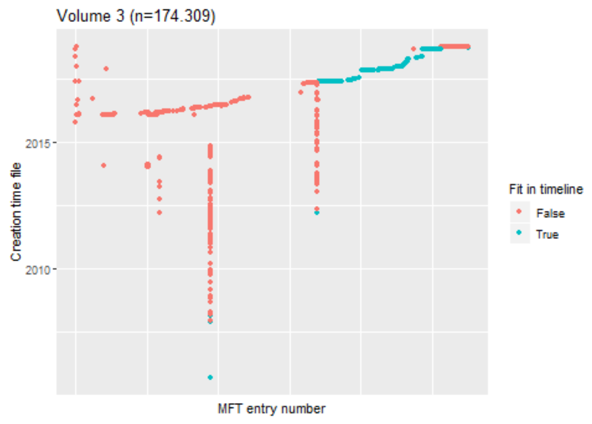


Figure 5.12: Volume 3 - all files.

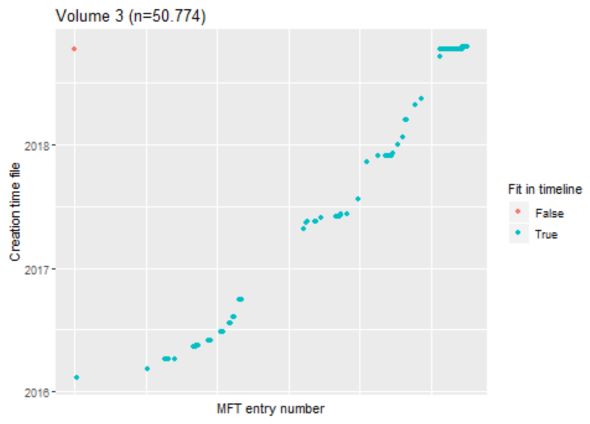


Figure 5.13: Volume 3 - created and copied files.

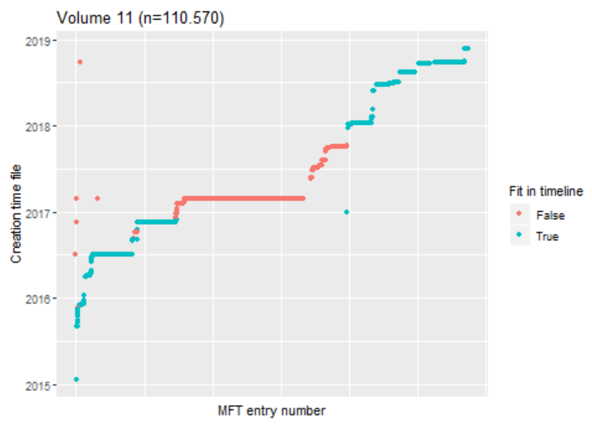


Figure 5.14: Volume 11 - all files.

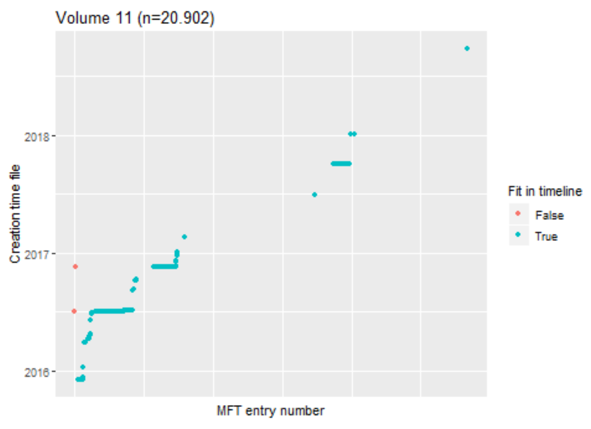


Figure 5.15: Volume 11 - created and copied files.

5.4 Fit Timeline and Filling Holes

Table 5.10 shows the number of files per volume that fits in a timeline with consecutive files having increasing creation times.

Table 5.10: Fit in timeline per volume, 12 volumes (n=581.458)

Volume	No. files fit	No. files no fit
1	13.997 (100%)	38 (0%)
3	80.713 (100%)	292 (0%)
4	82 (93%)	6 (7%)
5	109.323 (100%)	32 (0%)
8	7.172 (96%)	271 (4%)
9	0 (0%)	2 (100%)
10	120.448 (100%)	74 (0%)
11	18.018 (83%)	3.684 (17%)
13	14.749 (99%)	82 (1%)
14	0 (0%)	2 (100%)
15	157.342 (99%)	1.071 (1%)
18	53.904 (100%)	156 (0%)

Figures 5.16 and 5.17 illustrate the above results for volumes 3 and 11. The files that fit in the timeline and not are distinguished by colour. The blue files are the files with increasing first block number having increasing creation times, and form a so called timeline based on location. The red files are the files which have a creation time later than the creation time of a file with a higher first block allocated.

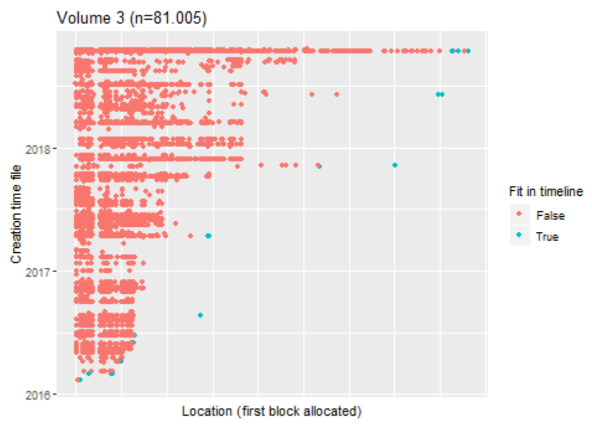


Figure 5.16: Volume 3 - timeline whole volume.

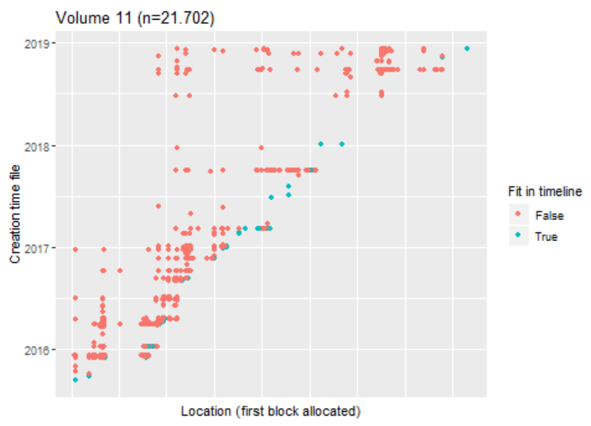


Figure 5.17: Volume 11 - timeline whole volume.

Table 5.11 shows the existence of holes after files and the relation between holes after files and the timeline. 16% of the files has a hole behind.

Table 5.11: Overall fit in timeline and holes behind files, 416 volumes (n=18.952.619)

	Hole behind file		
Fit in timeline	Yes	No	Total
Yes	24.742	258.385	283.127
	9%	91%	1,5%
	1%	2%	
No	3.088.528	15.580.964	18.669.492
	17%	83%	98,5%
	99%	98%	
Total	81.716	499.742	581.458
	16%	84%	

5.5 Partial Timelines

Partial timelines are created for all volumes. The volumes are divided in parts of 1.000 files and in parts of 100 files. Table 5.12 shows the number of parts per volume.

Table 5.12: Number of volume parts per volume

Size volume part	No. of volumes	Total no. of parts	Number of volume parts		
			Minimum	Mean	Maximum
1.000 files	272	18.810	1	69	669
100 files	295	189.362	1	642	6.691

Per partial timeline the number of files that fits in the timeline is counted. The overall results are described in Table 5.13.

Table 5.13: Number of files that fits in timeline, per volume part (parts of 1.000 files: n=18.810, parts of 100 files: n=189.362)

Size volume part	Number of files that fits in timeline			
	Minimum	Mean	Maximum	Median
1.000 files	1	79	1.000	14
100 files	1	20	100	9

Figures 5.18 and 5.19 show individual timelines for volume parts that consist of 1000 files, and Figures 5.20 and 5.21 for volume parts of 100 files.

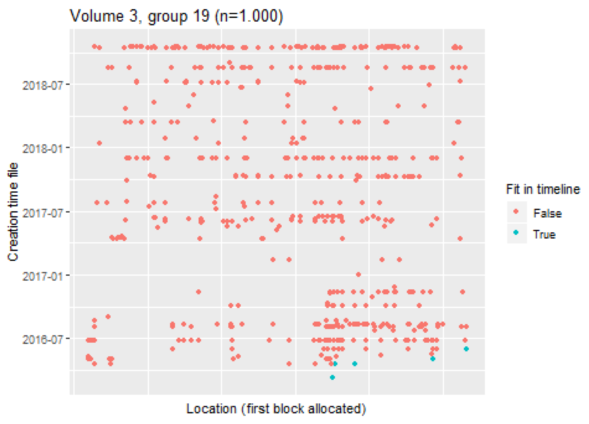


Figure 5.18: Timeline volume 3, group 19 (n=1000).

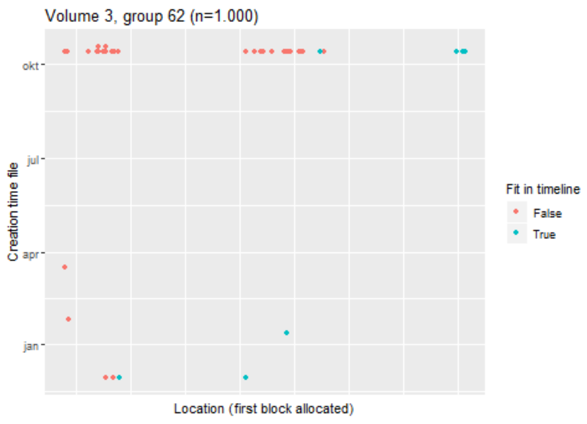


Figure 5.19: Timeline volume 3, group 62 (n=1000).

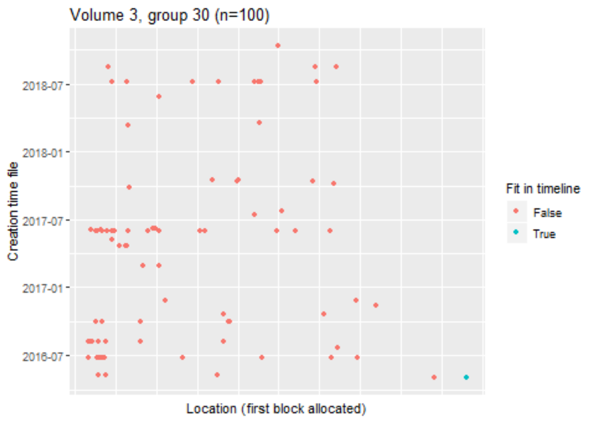


Figure 5.20: Timeline volume 3, group 30 (n=100).

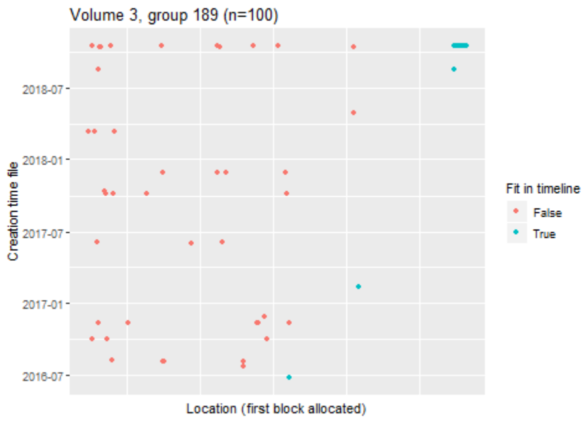


Figure 5.21: Timeline volume 3, group 189 (n=100).

5.6 Derivation Creation Time

Table 5.14 shows, for 4 variants of the partial timeline strategy and for the 10 neighbours strategy, the minimum, mean, maximum and standard deviation of the deviation of the derived creation time versus the actual creation time, as observed in all included volumes in the Wildfrag database. As the files that are placed consecutively and the files that are placed between existing files are not distinguished, the methods are applied on both type of files.

Table 5.14: Deviations derived creation times

Method	Calculation method	No. of files	Minimum (days)	Mean (days)	Maximum (days)	Standard deviation (days)
"Partial timeline", 1.000 files per part	Next crtime	12.248.687	-1.661	3	16.355	181
	Mean previous and next crtime	12.248.687	-9.728	-108	8.178	205
"Partial timeline", 100 files per part	Next crtime	14.823.318	-1.668	3	17.362	156
	Mean previous and next crtime	14.823.318	-9.629	-64	8.681	160
"10 neighbours"	Mean creation times previous 5 and next 5 files	18.949.129	-2.537	0	17.705	116

In Table 5.14 it is visible that the standard deviation of the deviation of the derived creation time versus the actual creation time is the lowest for the 10 neighbours strategy. Figure 5.22 shows the degree of deviation, after applying the 10 neighbours strategy, of 30 volumes, as boxplots.

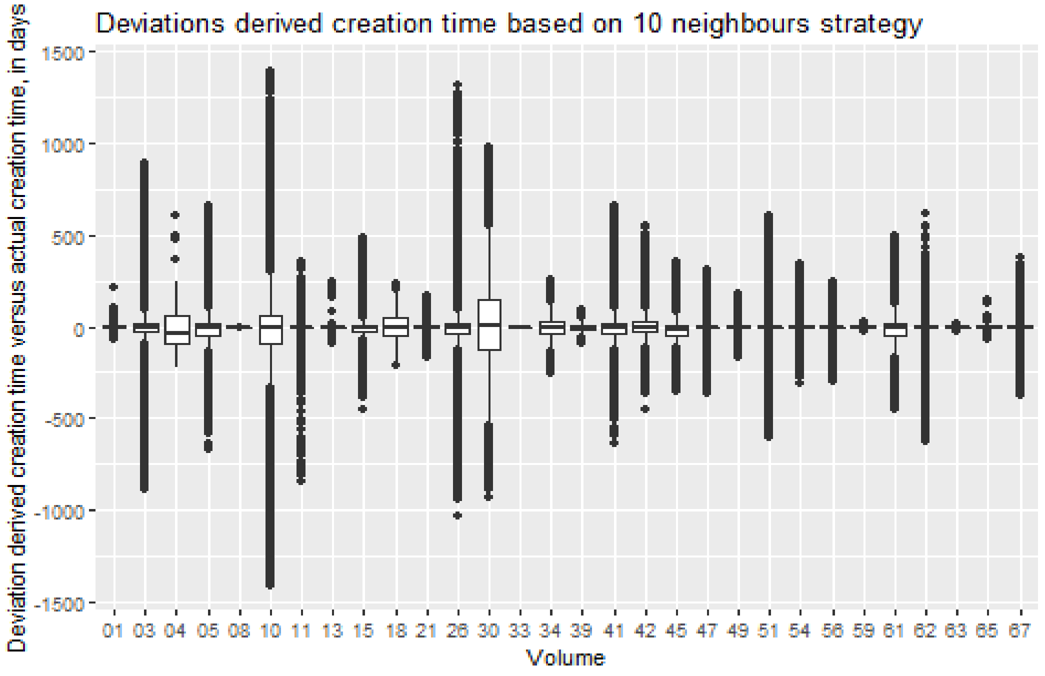


Figure 5.22: Boxplots deviations derived creation time versus actual time, per volume (n=30).

Table 5.15 contains an overview of all standard deviations of the deviations of the derived creation time versus actual time, divided in categories, after applying the 10 neighbours strategy. 166 volumes (52%) have a standard deviation ≤50 days. For 3 volumes the standard deviation is not calculated because these volumes contain only 1 file, besides the 10 neighbours.

Table 5.15: Distribution standard deviation per volume of deviations derived creation time versus actual time (n=320)

Standard deviation	Frequency	Percentage
<5	34	11%
5-<10	23	7%
10-<25	55	17%
25-<50	54	17%
50-<100	67	21%
100-<200	64	20%
200-<300	11	3%
$\geq$ 300	9	3%
NA	3	1%

Figures 5.23 and 5.23 show the deviations of the derived creation time versus the actual time of 2 volumes, after applying the 10 neighbours strategy. Note that the y-axes have a completely different scale.

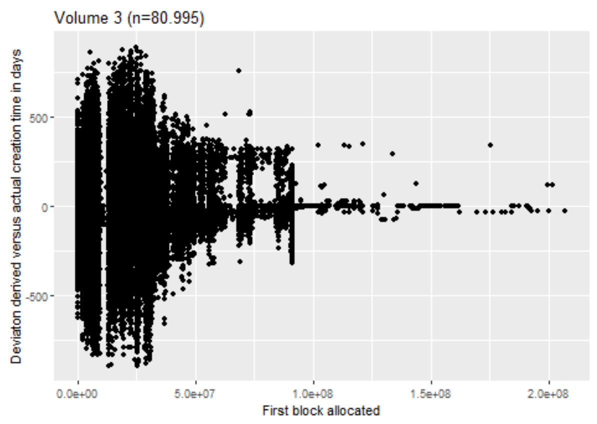


Figure 5.23: Deviations derived creation time, based on 10 neighbours strategy, volume 3 (n=80.995).

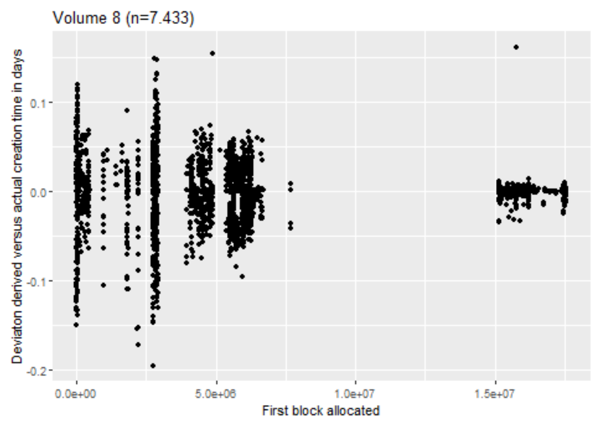


Figure 5.24: Deviations derived creation time, based on 10 neighbours strategy, volume 8 (n=7.433).

5.7 Derivation Deletion Time

In Section 4.4 a strategy is described to determine the latest time a file was not deleted. Table **??** contains examples of 4 files. 2 files are small and 2 files are big. The table also contains both files with a hole after the file and files without. On the left side of the table the original file is described. The right side of the table contains the information of the file that is used to determine the latest time the file was not deleted. This is the latest file with a size equal to or smaller than the size of the original file, and with a space bigger than the space used by the original file. (As mentioned in Section 4.4 the space of a file consists of the number of blocks allocated for the file and the number of unallocated blocks after the file.)

Table 5.16: Derivation of latest time not deleted for 4 files

File_id	Start block	End block	Next start block	No. of blocks allocated	No. of blocks hole	Total space	Creation time	File_id	No. of blocks allocated	No. of blocks hole	Total space	Latest time not deleted
112098	35	36	37	2	0	2	16-10-2017 17:27	336149	2	67	69	24-10-2018 09:01
112150	204	204	208	1	3	4	11-8-2018 06:48	336149	2	67	69	24-10-2018 09:01
116667	24620	25698	25699	1079	0	1079	26-1-2014 17:07	333530	1	2142	2143	24-10-2018 08:48
162777	935589	936662	936677	1074	14	1088	31-1-2016 14:40	333530	1	2142	2143	24-10-2018 08:48

5.8 Analysis of the Results

5.8.1 Relation Allocation of Blocks and Creation Time of a File

Methods that use the location of a file for file dating are based on the idea that files are written in a structured way. The most obvious idea is that, on a disk, the first, low numbered blocks are written before the blocks beyond. If this is done consistently, then it is expected that most of the blocks in the first part of the volume are allocated and most of the blocks in the last part of the volume are unallocated, depending on the occupation rate of the volume. The graphs 5.8 and 5.9 show that, despite an occupation rate lower than 50%, still a lot of blocks with high numbers are allocated. This can partly be explained by the allocation of blocks with higher numbers, followed by the deletion of files with lower numbered blocks. Also the blocks allocated for the MFT are not visible in these graphs. Besides that, the lowest available blocks are not always allocated too.

Another complication of file dating based on the physical location of the file are the file operations that change the allocated blocks or the creation time. It is assumed that this effect is eliminated by only taking into account the files that are a result of the file operations "Create" and "Copy". This is tested with the timeline of the MFT entries, because this timeline is probably only disturbed by file operations, and the timeline based on the physical location is at least also disturbed by files placed between existing files. The tested hypothesis is: "if the MFT entry is used for the first time and only the file operations "Create" and "Copy" are included, then increasing MFT entries have increasing creation times". This is true for the majority of the volumes in the Wildfrag database. Therefore only the "Create" and "Copy" file operations are used to exclude the disturbing effect of the file operations.

What else can the created timeline based on MFT entries be used for? This timeline cannot be used to derive a creation time in this research, because starting point in this research is that the metadata of a file is not available. However, if the MFT entry and sequence number are available, the MFT entry is used for the first time, and the creation time cannot be trusted, then this method may be used to derive or validate the creation time.

Third main complication of file dating based on the physical location of the file are files placed between existing files, to fill holes that are arisen because of the deletion of files. As long as files are placed consecutively, a clear timeline with consecutive files having increasing creation times exists. In practice, the amount of files that fits in the timeline is not substantial, see section 5.11. It is assumed that the files that do not fit in the timeline are placed between existing files, so in practice lots of files are placed between existing files.

As described in Section 5.5 the procedure to select the files that fit in the timeline for a whole volume is strict. Therefore partial timelines consisting of 100 and 1000 files are created, which only reflect a part of a volume. The results in Section 5.5 show that huge differences exist between the various partial timelines. The number of files that fits in the timeline varies from 1 to 1000 in case of parts of 1000 files, and from 1 to 100 in case of parts of 100 files. The median values are 14 and 9, so most of the parts contain a relatively small part of the files that fits in the timeline. As partial timelines vary a lot, no general characteristics can be derived.

5.8.2 Derived Creation Time

To derive a creation time, it is important to know if a file is placed consecutively or between existing files, because only if it is placed consecutively the creation time can be derived from the timeline with consecutive files having increasing creation times. An assumption is that files that fit in the timeline only sometimes have holes behind, and files that do not fit often have holes behind. In practice in the Wildfrag database (see section 5.11), 9% of the files that fits in the timeline has holes behind, but only 17% of the files that do not fit in the timeline has holes behind. A large part of the files that does not fit in the timeline, 83%, has no holes behind. Therefore the holes behind files are not a good indicator to distinguish the files that fit in the timeline and the files that do not.

In this research the partial timeline strategy and the 10 neighbours strategy are used to derive creation times and the results are compared, see Section 5.6. In the situation with files that are placed between existing files that cannot be recognised, the 10 neighbours strategy appears to be a better method to derive a creation time of a file. The mean deviation is 0 and the standard deviation of this method is lower than the standard deviations of the other methods. This means that a more precise creation time can be derived with this method.

The mean standard deviation of the deviations between the derived and actual creation times for the 10 neighbours strategy is substantial. The graphs in Section 5.6 show that large differences exist in the degree of deviation between the volumes. Therefore the calculated overall mean and standard deviation, as presented in table 5.14 cannot be applied to individual volumes.

For individual volumes the mean, standard deviation and a 95% confidence interval of the deviations are derived, see the examples in Table 5.17. If the upper limit of the confidence interval is added to the derived creation time of the concerning file, then this results in a time with 97,5% certainty that the file is created before that time.

Table 5.17: Examples statistics deviations derived creation time versus actual creation time

Volume	Mean (days)	Standard deviation (days)	95% confidence interval
8	1,2	0,02	(1,16; 1,24)
13	-9,6	7,5	(-24,6; 5.4)
3	0,0	185	(-370; 370)
10	3,1	306	(-609; 615)

This method makes sense if the standard deviation is not too big. It is not clear which standard deviations satisfy in practice. The results of volumes 3 and 10 in Table 5.17 are considered as not practical but, in Table 5.15 is shown that 52% of the volumes has a standard deviation ≤50 days.

This approach results in a creation time with a certainty of 97,5% that a file is created before that time, and not 100%. If 97,5% certainty can be reached for more than 1 file then the joint certainty increases. For 1 file the chance a file is created after that time is 0,025, for 2 files $0{,}025^2$=0,000625 and for 3 files $0{,}025^3$=0,000015625.

Implication of these results for practice is that, to derive a 97,5% certainty creation date, first the volume specific statistical distribution of the deviations needs to be determined, see Figure 5.25.

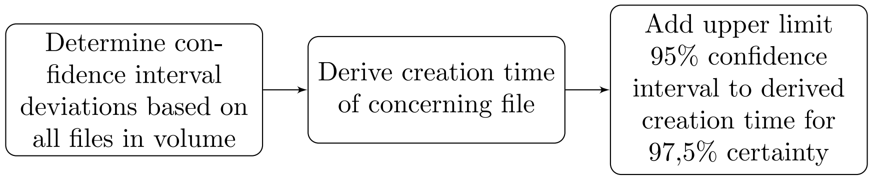

Determine confidence interval deviations based on all files in volume
Derive creation time of concerning file
Add upper limit 95% confidence interval to derived creation time for 97,5% certainty

Figure 5.25: Steps to derive a creation time with 97,5% certainty

5.8.3 Derived Deletion Time

Table 5.16 contains 4 examples of the derivation of the latest time a file is not deleted, of files of volume 1. The data collection time of this volume is: 24-10-2018 11:04:22. Assumed the files are deleted at the data collection time, the derived latest times the files were not deleted, are just before this virtual deletion time.

The applied method is based on the best fit allocation strategy. According to this strategy, files are placed in the smallest possible hole. The last 2 examples show a file with a size of 1 block, placed in a hole of 2142 blocks. In theory it is possible that, at that moment, no smaller hole existed, but it seems best fit is not applied here. This cannot be verified with the current data, because the Wildfrag database does not contain deleted files.

Other factors that determine the allocation of files besides the best fit strategy are not known at this moment. To further develop the method to derive a deletion time, other factors that affect the allocation of files needs to be investigated and implemented in the method.

Chapter 6

Discussion

6.1 Timelines based on MFT Entries

The data of the Wildfrag database used to create the timelines based on MFT entries, contains some observations with duplicate MFT entry numbers within a volume. As the MFT entry number is unique within a volume this should not be possible. Only few duplicate observations exist so the impact is only small but it would be better to exclude observations with duplicate MFT entry numbers from this analysis.

To create the timelines based on MFT entries, shown in section 5.8.1, only files that have blocks allocated are included. In this research the location of files, determined by the blocks allocated is relevant, and the analysis of the timelines based on MFT entries is only supportive. As mentioned in section 5.8.1, the method described may be used to derive a creation time if the MFT entry and sequence number of a file are known and the MFT sequence number is 1. In that case, there is no reason to only select the files that have blocks allocated. Also resident files and symbolic links may be used as a part of the timeline.

For this research, that focuses on the location and creation times of files, the MFT entry numbers offer no extra information. Files that are a result of the file operations ”Create” and ”Copy”, and with a MFT entry that is not reused yet, show a strong correlation between the creation time and the MFT entry number. If files are ordered by creation time, then they are also, for the most part, ordered by MFT entry number, and the other way around. In this research the relation between location and creation time is analysed. If the creation time is known then the MFT entry number does not contribute anything extra.

6.2 Factors that affect the degree of deviation per volume

As mentioned in section 5.8.2, the degree of deviation of derived creation times versus actual creation times varies considerably between volumes. The question is what causes these differences. In section 3.1 is already described that, according to Casey, allocation patterns in actively used NTFS systems are not predictable because the system makes efficient use of the storage space [6]. It is not clear how to determine if a system is actively used or not. Table 6.1 shows the relation between the number of files per volume and the degree of deviation. They are not strongly correlated. Other factors that might affect the degree of use of the system are: the occupation rate and the period the system is in use. According to Bahjat the start of the use of the system can be derived from the creation date of the first record, which is the $MFT file itself [4].

Knowledge on these factors might help to judge the applicability of the described methods in practical situations.

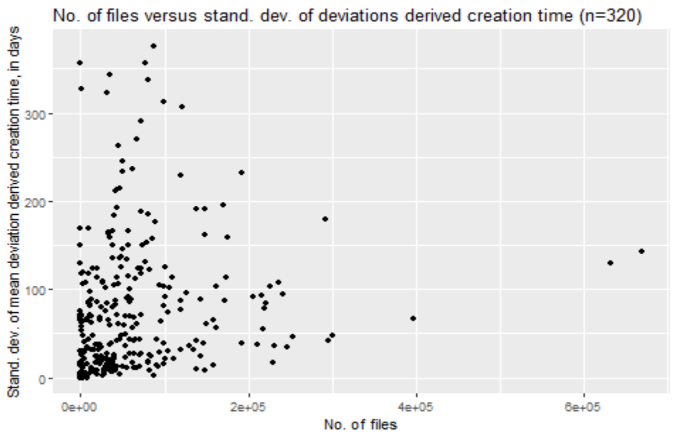


Figure 6.1: Number of files per volume versus standard deviation mean deviation derived creation time versus actual time (n=320).

6.3 Validation of Results

During this research only creation times are derived for files that are a result of a ”Create” or ”Copy” file operation because other files have a creation time that probably does not reflect the actual creation time. Therefore it was not possible to test files which are a result of other file operations. It is assumed that the applied methods also work for those files. This should be verified by experiments in which other file operations are performed, while saving the original creation time.

Also, the Wildfrag database does not contain deleted files. This is not important to test the derivation of a creation time because the derivation of a creation time can be tested with files that are not deleted. To test the derivation of a deletion time, it is assumed that an arbitrary file is deleted just after the data collection time. However, with the data in the Wildfrag database it is not possible to verify that a derived latest time a file was not deleted is the actual latest time the file was not deleted. Experiments are required with files that are really deleted, to validate the method and, probably, the method needs to be further developed. If the method proves to be correct then it may be applied to all files in the Wildfrag database to determine the reliability.

6.4 Reflection on Data Analysis with R

The experienced advantages and disadvantages of the programming language R to perform the analyses for this project are described below.

Advantages R:

1. R consists of a large amount of packages that facilitates programming. Especially the “Tidyverse” is a package that makes programming in R easy and guarantees a quick start.

2. R is an open source tool. Therefore new packages and improvements are implemented constantly.

3. R is a functional language and functions are first class values. This makes it possible to break down a programming task into subtasks and to write compact code.

4. The basic data structure in R is the vector. Generally functions are applied to all elements in a vector and no iteration on the elements of the vector is required.

5. R is dynamically typed. When different types are used interchangeably, implicit coercion happens to the most complex type. This works intuitively and makes programming flexible.

6. R contains a lot of possibilities to get other types of data into R. For this project it has been very useful to run SQL queries against a SQLite database, that returned R data frames, because the Wildfrag database is a SQLite database. It is also possible to create a SQLite database from R, for persistent storage and this is also used during the project.

7. R contains a lot of possibilities to create output, from tables and graphs to dashboards. For example, during this project the possibility to save a created table in the form of Latex code was very useful.

8. R is good at the visualisation of data. The creation of graphs with R is quite accessible, it is easy and it does not take much time. To create a first graph in R, only a few settings are required because R uses a lot of defaults. For example, the axis are determined based on the values in the data. On the contrary, a lot of options are available to fully customise a graph.

   Disadvantages R:

1. Because of all the packages and versions it is sometimes hard to get a good overview of the packages and to get access to the most up to date versions.

2. Also because of the huge number of packages, a lot of redundancy exists. Several functions and solutions exist to perform the same kind of action.

3. R is both a functional and an object oriented language. The extended possibilities of R complicate the learning of the programming language R.

4. R is very fast when working with huge data sets, as long as tasks are handled vector wise. If a programming task cannot be handled vector wise then the task requires much more time.

5. All data that is read and written in R is kept in memory. Therefore R is a profligate user of memory. When working with huge data sets, the memory of the used system is limiting to perform analyses.

The conclusion is that R is an appropriate and convenient tool to perform analyses in a project like this, if and only if the system that is used has enough memory that matches with the amount of data.

Chapter 7

Conclusions

Complications to derive a creation time based on the physical location of a file are:

1. In practice the blocks with the lowest numbers are not always allocated before blocks with higher numbers.
2. Several file operations disturb a timeline with consecutive files having increasing creation times, because they change the creation time or the allocated blocks of a file.
3. After the deletion of files, holes arise between existing files, and these holes are filled with new files.

The research has shown that the effect of the file operations is excluded by only taking into account the files that are a result of the "Create" and "Copy" file operations as references to derive a creation time. It has not been possible to distinguish files that are placed consecutively and files that are placed between existing files. If this would be possible then, probably, the partial timelines approach would be a good method to derive a creation time for a consecutive placed file. As this is not possible, the existing 10 neighbours strategy proves to be the best method to derive a creation time on systems with Windows 10, with file system NTFS and block size 4096.

Usage of the 10 neighbours method results in big differences between volumes with respect to the deviations of the derived creation time versus the actual creation time. Because of these differences it is not possible to make generally valid statements. Further research is needed to clarify the differences between the volumes.

IF the deviations of the derived creation time versus the actual creation time for a volume are quite small then it makes sense to use the method for that volume. Then it is possible to determine with a certainty of 97,5% that a file is created before the derived creation time plus 2 times the standard deviation. Also, the reliability may be increased by the investigation of more files.

The developed method to determine the latest time a file was not deleted, is completely based on the best fit allocation strategy but, in examples of the Wildfrag database, best fit seems not applied. Experiments with deleted files are required to further develop this method.

Chapter 8

Future Work

8.1 Improvement of Methods Creation and Deletion Time

The methods developed in this research are based on the assumptions that the lower blocks on a disk are allocated before the higher blocks and that the best fit strategy is applied. It has become clear that these factors do not fully explain the allocation of files but also other factors have impact. In this research, a more statistical approach to derive a creation time proved to be better than an approach based on the file allocation theory. If it becomes clearer how file allocation works, then the methods based on the file allocation theory can be improved.

The effect of the file type on the allocation of files can still be looked into in the database. Maybe different user applications result in different file application strategies. Probably better results will be achieved faster if the analysis of the database is combined with performing experiments. As the Wildfrag database contains snapshots of systems, the order in which the file operations are performed is not exactly clear. By doing experiments, the status before and after a specific action can be recorded. The WildFragSim tool as developed by Robbert Noordzij might be used for this purpose.

Experiments to be performed should focus on the allocation mechanisms of files. If the status of a disk can be kept before and after a file operation, then it is possible to determine if the best fit allocation strategy is applied. Files with the best fit strategy applied and not should be compared. For example, some files might reserve space to extend. Is it possible to recognise these files? And is it possible to recognise files that are placed between existing files? Also the filling of a disk over time can be followed with experiments.

After improvement of the method to derive a creation time, the Wildfrag database can be used to test the method. The Wildfrag database has added value here, because it contains lots of files, collected from real users. To test the method to derive a deletion time, the Wildfrag database cannot be used, because it does not contain a deletion time. To validate this method, also data from real users, containing the deletion time, should be collected.

8.2 Related Issues

As mentioned in Section 3.5, defragmentation results in an underestimation of the possession of a file. How often this occurs should be investigated, to properly assess this effect.

In this research, after applying the 10 neighbours strategy, the statistical distribution of the deviations, which are a result of the comparison of the derived and actual creation time, appeared to differ largely between volumes, see section 6.2. The factors that might cause these differences, like the occupation rate of the volume, the number of files on a volume and the life span of the system should be investigated. If it becomes clearer what causes the deviations, then also this method can be improved.

During the research several data issues came across, like incomplete date values, date values in the future and MFT entries within a volume, that are not unique. These issues should be clarified and the effect on the results as described in this research should be investigated.

Bibliography

[1] S. Raghavan, "Digital forensic research: current state of the art," *CSI Transactions on ICT*, vol. 1, no. 1, pp. 91–114, 2013.

[2] B. Carrier, *File system forensic analysis.* Addison-Wesley Professional, 2005.

[3] A. Silberschatz, P. B. Galvin, and G. Gagne, *Operating Systems Concepts.* Wiley, 2014.

[4] A. A. Bahjat and J. Jones, "Deleted file fragment dating by analysis of allocated neighbors," *Digital Investigation*, vol. 28, pp. S60–S67, 2019.

[5] R. Sears and C. van Ingen, "Fragmentation in large object repositories," *arXiv preprint cs/0612111*, 2006.

[6] E. Casey, "Digital stratigraphy: Contextual analysis of file system traces in forensic science," *Journal of forensic sciences*, vol. 63, no. 5, pp. 1383–1391, 2018.

[7] A. Pal and N. Memon, "The evolution of file carving," *IEEE signal processing magazine*, vol. 26, no. 2, pp. 59–71, 2009.

[8] S. L. Garfinkel, "Carving contiguous and fragmented files with fast object validation," *digital investigation*, vol. 4, pp. 2–12, 2007.

[9] K.-P. Chow, F. Y. Law, M. Y. Kwan, and P. K. Lai, "The rules of time on ntfs file system," in *Systematic Approaches to Digital Forensic Engineering, 2007. SADFE 2007. Second International Workshop on.* IEEE, 2007, pp. 71–85.

[10] J. Bang, B. Yoo, and S. Lee, "Analysis of changes in file time attributes with file manipulation," *digital investigation*, vol. 7, no. 3-4, pp. 135–144, 2011.

[11] X. Ding and H. Zou, "Time based data forensic and cross-reference analysis," in *Proceedings of the 2011 ACM symposium on applied computing.* ACM, 2011, pp. 185–190.

[12] T. Sharma and M. Kaur, "Time rules for ntfs file system for digital investigation," *International Journal of Advanced Research in Computer Engineering & Technology (IJARCET), ISSN*, pp. 1146–1151, 2015.

[13] G.-S. Cho, "A computer forensic method for detecting timestamp forgery in ntfs," *Computers & Security*, vol. 34, pp. 36–46, 2013.

[14] G. S. Cho, "An intuitive computer forensic method by timestamp changing patterns," in *2014 Eighth International Conference on Innovative Mobile and Internet Services in Ubiquitous Computing (IMIS).* IEEE, 2014, pp. 542–548.

[15] T. Knutson and R. Carbone, "Filesystem timestamps: What makes them tick," *GIAC GCFA Gold Certification*, vol. 11, 2016.

[16] J. R. Douceur and W. J. Bolosky, "A large-scale study of file-system contents," *ACM SIGMETRICS Performance Evaluation Review*, vol. 27, no. 1, pp. 59–70, 1999.

[17] N. Agrawal, W. J. Bolosky, J. R. Douceur, and J. R. Lorch, “A five-year study of file-system metadata,” *ACM Transactions on Storage (TOS)*, vol. 3, no. 3, p. 9, 2007.

[18] D. T. Meyer and W. J. Bolosky, “A study of practical deduplication,” *ACM Transactions on Storage (TOS)*, vol. 7, no. 4, p. 14, 2012.

[19] A. Field, J. Miles, and Z. Field, *Discovering statistics using R.* Sage publications, 2012.

[20] H. Wickham and G. Grolemund, *R for data science: import, tidy, transform, visualize, and model data.* ” O’Reilly Media, Inc.”, 2017.

Appendix A

Description Variables Wildfrag Database

Table A.1: Systems table

Nr	Name column	Description	Type column	Derivation
1	id	Sequential number identifier. Primary key of this table.	Integer	Generated, unique identifier for each row.
2	start_run	Start-timestamp of the datacollection[1]	DateTime	
3	os	Name of the operating system (OS) on the computer that executes the capture of data.	Text	Visual observation and manual entry[1]

Table A.2: Storage devices table

Nr	Name column	Description	Type column	Derivation
1	id	Sequential number identifier. Primary key of this table.	Integer	Generated, unique identifier for each row.
2	system_id	Foreign key to the System table[1]	Integer	
3	rotational	If the storage device is a hard disk drive or a solid state drive.	Boolean	True for HDD's, False for SSD's and other flash based storage devices.[1]

Table A.3: Volumes table

Nr	Name column	Description	Type column	Derivation
1	id	Sequential number identifier.	Integer	
2	storage_device_id	Foreign key to storageDevices table.	Integer	
3	fs_type	Identifier representing the filesystem on the partition or target volume.[2]	Text	Sleuthkit
4	size	Contains the size (in bytes) for the volume that is the target of analysis.[2]	Integer	Sleuthkit

[1]Information delivered by Vincent van der Meer, who created the database

[2]`https://docs.google.com/spreadsheets/d/1aiVUMP18H3eg5cUu-3y7fiKxdvVc4Jf7YIKRh5s5ZEQ/pub?output=html`

5	block_size	The size (in bytes) of an individual block of data in a volume, as defined by the file system. Block size varies with the size of the disk and the operating system. The block size is the minimum unit used by the operating system to store information on the disk. Knowledge of the block size in the target volume allows to determine the beginning of a file fragment.[2]	Integer	Sleuthkit

Table A.4: Files table

Nr	Name column	Description	Type column	Derivation
1	id	Sequential number identifier assigned by fiwalk/DFXML to individual files.[2]	Integer	Sleuthkit
2	volume_id	Foreign key to the Volumes table.[1]	Integer	Sleuthkit
3	mtime	The file's last modification time.[2]	Text	Sleuthkit
4	ctime	The file's entry modification time.[2]	Text	Sleuthkit
5	atime	The file's access time.[2]	Text	Sleuthkit
6	crtime	The file inode's creation time.[2]	Text	Sleuthkit
7	num_blocks	The number of blocks that are associated with a file.[1]	Integer	Derived from the allocated blocks.[1]
8	fragmented	If a file is fragmented. This can be forward fragmented or backward fragmented.	Boolean	True if the individual parts are or are not in consecutive block-increasing order.[1]
9	hardlink_id	Unique over the entire File table (and not just unique per Volume). Gives each hardlinked file a hardlink-ID. When a file has N hardlinks, you'll find N+1 identical hardlink'id.[1]	Integer	Derived from the allocated blocks.[1]
10	resident	True if the file is a resident file.[1]	Boolean	Sleuthkit
11	fs_seq	Incremental file number for entries in NTFS filesystems.[2] This is the sequence number of the Master File Table.	Integer	Sleuthkit
12	fs_inode	The filesystem assigns a uniquely identifiable inode number to each file in the filesystem.[2]	Integer	Sleuthkit

Table A.5: Blocks table

Nr	Name column	Description	Type column	Derivation
1	id	Sequential number identifier for each block that is available.	Integer	
2	file_id	Foreign key to the Files table.	Integer	Sleuthkit
3	startblock	The derived first block that is associated with a file.	Integer	
4	endblock	The derived last block that is associated with a file.	Integer	

Appendix B

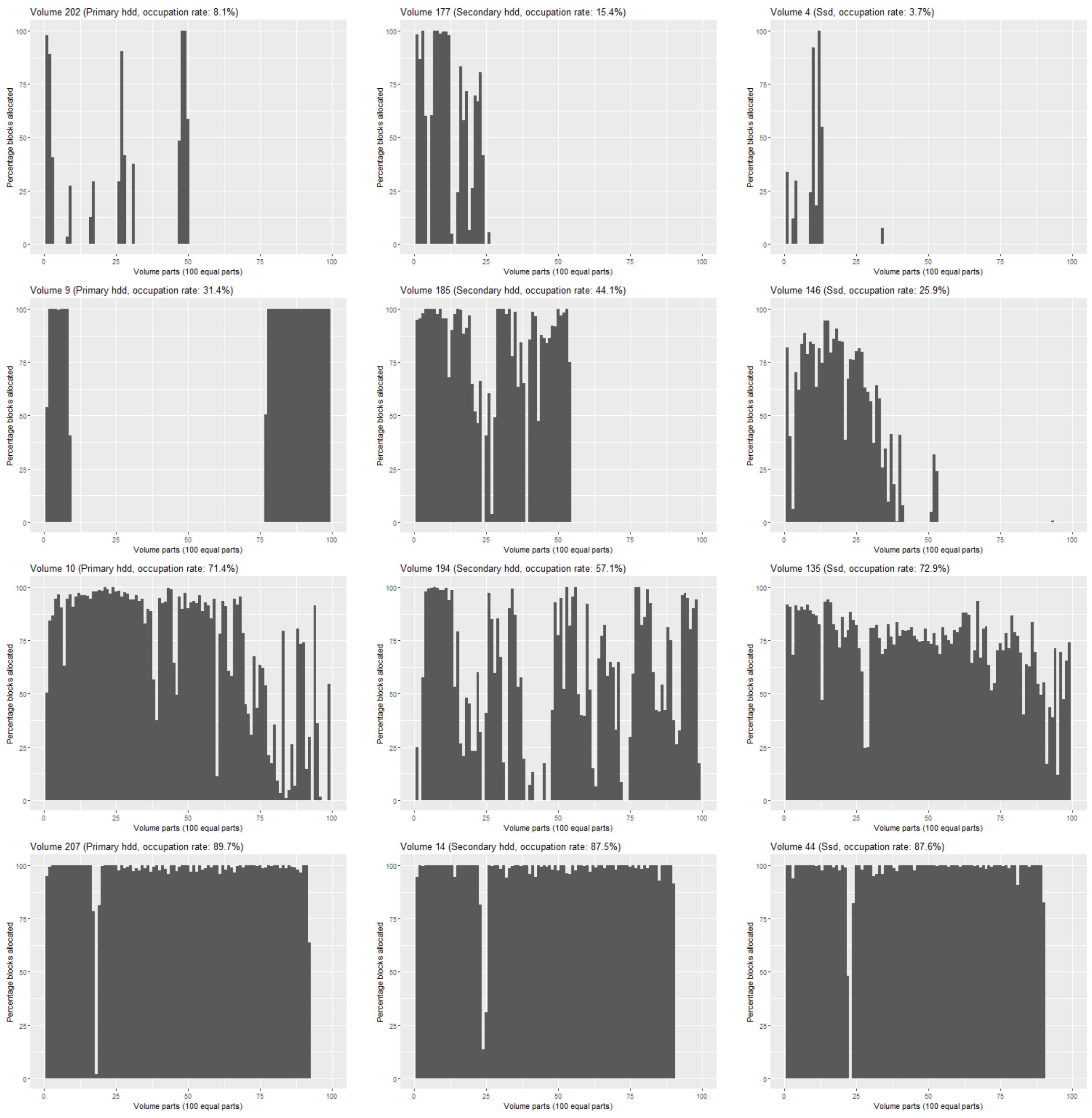
Validation checks

Table B.1: Overview performed validation checks

Nr	Check	Description	Result
1	Consistency systems, disks, volumes and files	Each system should have at least 1 disk, each disk should have at least 1 volume and each volume should have at least 1 file.	Checked, as expected
2	Number of files per system	It is expected that systems should have at least >250 files.	5 systems contain $\leq$ 250 files. These systems are excluded.
3	Number of files per volume	Volumes should consist of a minimum number of files. For this research at least 10.	The research dataset contains 250 volumes with <10 files. These volumes are excluded from the analyses.
4	Occupation rate versus total disk space	The occupation rate should be any value between 0 and 100.	Checked, as expected
5	Timestamps in future	Time stamps in future do not reflect the actual time the file operation is performed.	The research dataset contains 72 creation times in the future. These are excluded from the analyses.
6	Missing or incomplete timestamps	Missing and incomplete timestamps cannot be interpreted as time values.	The research dataset contains 18.399 missing or incomplete creation times. These are excluded from the analyses.
7	Blocks for files that are not fragmented	Files that are not fragmented should not have more than 1 set of blocks allocated.	Checked, as expected
8	Blocks for fragmented files	Files that are fragmented should have more than 1 set of blocks allocated.	Checked, as expected
9	MFT entry number	The MFT entry number should be unique within a volume.	The research dataset contains double MFT entry numbers per volume but only 0.003% of the files is involved. This aspect is not important for the analyses, therefore these values are ignored.

Appendix C

Location of files per volume



Appendix D

Reflection on Process

During the project I experienced that problems that seem quite simple are always more complex than you think. It helped to start to write them down and work them out in a structured way. During this writing process it turned out to be difficult to put ideas into words, and to take along the reader in the research area. I think it is in general important to be able to transfer knowledge, so, to make my thoughts clear, was a challenge in itself for me. The remarks of the supervisors helped a lot to improve my texts. I experienced that only small remarks are enough to bring the texts further, and to inspire me. I really appreciate the feedback and support I received from my supervisors along the way.

Before the project I was unfamiliar with the programming language R. Programming with an unknown programming language felt like a handicap. Especially in the beginning, I knew what to do but not how to program it. During the project I have really developed my R skills. I think it is only possible to learn a programming language by using it. In addition, during the project I attended the course Concepts of programming languages too. Also this theoretical framework helped me to get a good overview of R. At the end I am glad that I have used a language I did not know before.

We worked in a team of 3 persons, with different backgrounds and varying experience. We worked as a team, but the considerable differences resulted in limited added value of the team work. It was nice to share insights on the content, because there are not so many people to share such specific knowledge with. Besides that, it would have been nice if it was possible to take more advantage of the team work.

My interest is mainly in the data science field. This project has learned me, concerning data analysis, that it is important to first understand the problem you are working on thoroughly. It does not make sense to do just some analyses, but the analyses must be focused. The problem needs to be worked out first and this should not be underestimated.

During this project, several times the results turned out to be different than I expected. For me it was quite interesting to view the results, and learn how it really works in practice. I liked the gaining insight from the data very much.

Finally, I would really like to thank my supervisors and team members for their contributions. I learned a lot, enjoyed it and I feel enriched by performing this project.