Towards a fully automated extraction and interpretation of tabular data using machine learning

Per Hedbrant
Abstract

Towards a fully automated extraction and interpretation of tabular data using machine learning

Per Hedbrant

Motivation
A challenge for researchers at CBCS is the ability to efficiently manage the different data formats that frequently are changed. Significant amount of time is spent on manual pre-processing, converting from one format to another. There are currently no solutions that uses pattern recognition to locate and automatically recognise data structures in a spreadsheet.

Problem Definition
The desired solution is to build a self-learning Software as-a-Service (SaaS) for automated recognition and loading of data stored in arbitrary formats. The aim of this study is three-folded: A) Investigate if unsupervised machine learning methods can be used to label different types of cells in spreadsheets. B) Investigate if a hypothesis-generating algorithm can be used to label different types of cells in spreadsheets. C) Advise on choices of architecture and technologies for the SaaS solution.

Method
A pre-processing framework is built that can read and pre-process any type of spreadsheet into a feature matrix. Different datasets are read and clustered. An investigation on the usefulness of reducing the dimensionality is also done. A hypothesis-driven algorithm is built and adapted to two of the data formats CBCS uses most frequently. Discussions are held on choices of architecture and technologies for the SaaS solution, including system design patterns, web development framework and database.

Result
The reading and pre-processing framework is in itself a valuable result, due to its general applicability. No satisfying results are found when using mini-batch K means clustering method. When only reading data from one format, the dimensionality can be reduced from 542 to around 40 dimensions. The hypothesis-driven algorithm can consistently interpret the format it is designed for. More work is needed to make it more general.

Implication
The study contribute to the desired solution in short-term by the hypothesis-generating algorithm, and in a more generalisable way by the unsupervised learning approach. The study also contributes by initiating a conversation around the system design choices.
Dedication

To Torbjörn.  

Thank you for your engagement and ambition. Thank you for continuously pushing me with high expectations. Thank you for having me twice in Taiwan.
To Thomas.

“Hä ä bar å´ åk´”
- Ingemar Stenmark
Contents

1 Introduction .................................................. 9
   1.1 Interpretation of spreadsheets .......................... 9
   1.2 The specific case ...................................... 10
   1.3 The organizations behind the study: CBCS and Nordron ........................................ 12

2 Background .................................................. 14
   2.1 Pre-study of commercial solutions ..................... 14
   2.2 Present research ....................................... 14
      2.2.1 The Table understanding problem .................. 15
      2.2.2 The Semantic web .................................. 16
      2.2.3 Other ............................................... 17
   2.3 Choosing approach ..................................... 17

3 Problem Description and Aim ............................... 19
   3.1 The desired solution ................................... 19
   3.2 The aim of this study ................................. 20

4 Theory .................................................. 21
   4.1 Spreadsheets ........................................... 21
      4.1.1 What is a spreadsheet? ............................. 21
      4.1.2 How to interpret spreadsheets ..................... 22
      4.1.3 Spreadsheets at CBCS ............................. 23
4.1.4 The preferred output format ........................................... 24

4.2 Data schemas and structures ............................................. 24

4.2.1 Definition of data structure ........................................... 24

4.2.2 Relational data model ................................................ 25

4.2.3 Semi-structured data ................................................ 26

4.2.4 Graph data .......................................................... 26

4.2.5 Tree-structured data ................................................ 26

4.2.6 Tree vs graph data ................................................ 27

4.2.7 Sequential data .................................................... 28

4.2.8 Unstructured data .................................................. 28

4.2.9 Big Data .......................................................... 28

4.3 Clustering techniques .................................................... 28

4.3.1 Linkage – Similarity measure between sets ........................ 29

4.3.2 Agglomerative clustering .......................................... 30

4.3.3 Mean shift clustering ............................................. 30

4.3.4 Felzenszwalb segmentation method ............................ 30

4.3.5 DBSCAN clustering ............................................. 32

4.4 Things to consider when clustering ................................... 32

4.4.1 The Curse of Dimensionality .................................... 32

4.4.2 The choice of similarity measure ................................. 33

4.4.3 Dimensionality reduction techniques .......................... 33

4.4.4 Clustering evaluation ........................................... 34

5 Method – The Unsupervised Learning Approach ....................... 36

5.1 Pre-processing .......................................................... 36

5.1.1 The intermediate matrix and primitive variables ............. 36

5.1.2 One Hot Encoding of categorical features .................... 37
<table>
<thead>
<tr>
<th>Section</th>
<th>Title</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>5.2</td>
<td>The input data</td>
<td>38</td>
</tr>
<tr>
<td>5.2.1</td>
<td>Assumptions on input files</td>
<td>38</td>
</tr>
<tr>
<td>5.3</td>
<td>Dataset A</td>
<td>38</td>
</tr>
<tr>
<td>5.4</td>
<td>Dimensionality reduction</td>
<td>39</td>
</tr>
<tr>
<td>5.5</td>
<td>Choosing learning model</td>
<td>40</td>
</tr>
<tr>
<td>5.6</td>
<td>Evaluation</td>
<td>41</td>
</tr>
<tr>
<td>5.6.1</td>
<td>Visualising the results</td>
<td>42</td>
</tr>
<tr>
<td>6</td>
<td>Result – The Unsupervised Learning Approach</td>
<td>43</td>
</tr>
<tr>
<td>6.1</td>
<td>Dimensionality reduction</td>
<td>43</td>
</tr>
<tr>
<td>6.1.1</td>
<td>Comparing Dimensionality reduction methods</td>
<td>43</td>
</tr>
<tr>
<td>6.1.2</td>
<td>The rank of the feature matrix</td>
<td>44</td>
</tr>
<tr>
<td>6.2</td>
<td>Clustering comparison</td>
<td>45</td>
</tr>
<tr>
<td>6.3</td>
<td>Only looking at sheets called Plate</td>
<td>47</td>
</tr>
<tr>
<td>6.3.1</td>
<td>Same files as before</td>
<td>47</td>
</tr>
<tr>
<td>6.3.2</td>
<td>Increased number of files</td>
<td>48</td>
</tr>
<tr>
<td>7</td>
<td>Method – The Hypothesis-generating Approach</td>
<td>50</td>
</tr>
<tr>
<td>7.1</td>
<td>Specialisation</td>
<td>50</td>
</tr>
<tr>
<td>7.2</td>
<td>Pre-processing</td>
<td>50</td>
</tr>
<tr>
<td>7.3</td>
<td>Hypothesis generation</td>
<td>52</td>
</tr>
<tr>
<td>7.3.1</td>
<td>Clustering</td>
<td>53</td>
</tr>
<tr>
<td>7.3.2</td>
<td>Cut the dendogram</td>
<td>55</td>
</tr>
<tr>
<td>7.3.3</td>
<td>Extracting relations</td>
<td>56</td>
</tr>
<tr>
<td>7.4</td>
<td>Hypothesis evaluation</td>
<td>58</td>
</tr>
<tr>
<td>7.5</td>
<td>Learning</td>
<td>60</td>
</tr>
<tr>
<td>8</td>
<td>Result – The Hypothesis-generating Approach</td>
<td>61</td>
</tr>
</tbody>
</table>
8.1 Agglomerative clustering ........................................... 63

9 Discussion ...................................................................... 66
  9.1 The Unsupervised Learning Approach ......................... 66
    9.1.1 Dimensionality Reduction ..................................... 67
    9.1.2 Evaluating the Clustering ..................................... 68
    9.1.3 Scaling ................................................................. 68
  9.2 The Hypothesis-generating Approach ......................... 68
    9.2.1 Learnings ............................................................. 69
  9.3 Future work .............................................................. 70

10 Conclusion .................................................................... 71

Bibliography ..................................................................... 72

A Key choices in design and technologies .......................... 79
  A.1 Overview of the system .............................................. 79
  A.2 System Design Patterns ............................................ 80
    A.2.1 REST ................................................................. 80
    A.2.2 Modularity .......................................................... 82
    A.2.3 Application Programming Interface ....................... 82
    A.2.4 Server-side Web API ............................................ 83
  A.3 Web Development Framework .................................... 83
  A.4 Web Server .............................................................. 84
  A.5 Interface between the Web server and the web application ........................................................................... 84
  A.6 Database ..................................................................... 85
    A.6.1 SciDB ................................................................. 85
    A.6.2 PostgreSQL .......................................................... 85
  A.7 Containerization ........................................................ 85
Chapter 1

Introduction

“In our experience, the tasks of exploratory data mining and data cleaning constitute 80% of the effort that determines 80% of the value of the ultimate data mining results. Data mining books (…) provide a great amount of detail about the analytical process and advanced data mining techniques. However they assume that the data has already been gathered, cleaned, explored, and understood.” — Dasu and Johnson (2003)

This study is in the field of data wrangling, the process of formatting data before the actual analysis can be done. The aim is to minimize the data analyst’s time spent on data preparation and cleaning, by creating a better understanding of the data schema in spreadsheets. In Section 1.1, we outline the problem of interpreting spreadsheets, in Section 1.2 we present a specific use case where the problem occurs, and in Section 1.3 we give an introduction to the organisations behind this study.

1.1 Interpretation of spreadsheets

Much of today’s knowledge on the Web and in print is kept in spreadsheets, tables and other structured documents (Syed, Finin, Mulwad and Joshi 2010). When using spreadsheets to store data, a human or a machine has few formal requirements to follow, but has instead the freedom and autonomy to create her own schema of the data. Therefore, spreadsheets have so called implicit schemas where metadata and data live together in the same tabular space (Bernardo, Santanchè and Baranauskas 2014). Compare this to a relational database where the schema is explicitly defined beforehand by the tables. Therefore, a relational database has formal and predefined rules on how to query data from it, something that a spreadsheet lacks. Generally, the flexible and implicit data schema of spreadsheets is designed for independent and isolated use, but makes the data difficult to merge with data from other sources (Bernardo et al. 2014).
1.2 The specific case

Thomas Lundbäck is a researcher at Chemical Biology Consortium Sweden (CBCS), working in the field of drug discovery. In his daily work, he conducts experiments in microtiter plates – all of them are in the sizes of 96, 384, or 1536 wells, see Figure 1.1. Many experiments are conducted in parallel, which requires efficient means for automated data import and transfer. Thomas adopts a broad range of different technologies from various manufacturers. A challenge he has is management of the plentiful and frequently changed data formats. This includes import of data and assignment of the correct plate maps, regardless of the instrument output. Today, this is done manually by Thomas and his fellow researchers, which is a time-consuming and error-prone process. Each time Thomas receives a new readout file from either a machine of his own or from an external collaborator, he needs to convert to his standard format before being able to analyse the data. This process adds no value at all to the end result, but constitutes a notable part of the time involved to reach it.

Figure 1.1: A multi-pipette and a microtiter plate with 96 wells.

There is a myriad of commercial solutions available for the analysis of High Throughput Screening (HTS) data, specifically addressed to researchers like Thomas. The most notable software products are PerkinElmer’s SciStream, CDD Vault Drug Discovery, and Genedata Screener. But, to our knowledge, all these solutions require data to be in a standard format. Some of them provide templates for the conversion process, but this still includes a large number of manual steps by the user. And, the format of the data must conform to the format of the given template. Any change in the structure of the file demands creation of a new template or editing the present one in order to load the data. When using SciStream for instance, the user must manually enter the number of rows to skip in the file header, how many rows and columns a block of data has, et cetera. And the template creation application still

---


2For more information, please visit [https://www.collaborativedrug.com/](https://www.collaborativedrug.com/) (retrieved 2018-11-02).

has strong constraints on the format of the data to begin with (Thomas Lundbäck, personal communication).

Figure 1.2: The machine output from a Victor 3 machine. The cells that need to be identified from the machine output are marked by coloured rectangles.

The raw data the researchers want to analyse comes from machines that measure the substrate in the microtiter plates. In this study, we use these files as input, and the aim is to interpret the schema and data structure of these files. These machines write the measurement values and their meta-data to Excel or comma-separated files. The types of machines considered in this study are Victor 2, Victor 3, Envision and Microbeta. We will call a file from one of these machines as being of file type Victor 2, Victor 3, Envision, or Microbeta.

The formats of the data commonly seen in these files are tables of 96, 384, or 1536 cells, with $8 \times 12$, $16 \times 24$, or $32 \times 48$ rows and columns. The Figures 1.2 and 1.3 show screenshots of two different sheets in the same file. The file is written by a machine called Victor 3. In this file, three $8 \times 12$ plates are found (highlighted in red), together with annotations above each block of data (some are highlighted in blue, yellow, and green). The data in the two sheets is identical, but is presented in two different formats. The key problem is to identify how the data in the different coloured squares relate to data in other coloured squares. Please read more in Appendix B about how the problem is handled at CBCS presently.
1.3 The organizations behind the study: CBCS and Nordron

This study is a collaboration between CBCS and Nordron. CBCS is a part of the Science of Life Laboratory (SciLifeLab) at Karoliska Institutet (KI), a national center for molecular biosciences established by KI, the Royal Institute of Technology, Stockholm University, and Uppsala University. A broad range of different analysis techniques for studies of biological responses are available at CBCS and its academic collaborators. CBCS is government-funded and non-profit. Its main strategic objective is to serve other researchers in the field of chemical biology by for example

4For more information, please visit [http://www.cbcs.se/](http://www.cbcs.se/) (retrieved 2018-11-02).
assay development and High Throughput Screening (HTS).

Nordron AB is a data analytics company founded in 2013. Nordron is specialised in identification of decision variables and deep learning for prediction.
Chapter 2

Background

In this chapter, we describe what has been done in this field before our work. In section 2.1 we present our findings of commercial or academic solutions. In section 2.2 we summarize the state of present research in the field. In Section 2.3 we discuss how to leverage on previous findings, and what approach to choose for this study.

2.1 Pre-study of commercial solutions

The need to automatically read and interpret data from files is not found at CBCS alone. A pre-study for this thesis was done to survey related commercial and academic software applications. The aim was (a) to identify any form of intelligent and automatic loading of data, and (b) to survey the file extensions supported by the applications. More than 50 software applications were identified in the fields of data analysis, statistics, data mining, data visualization and scientific computing. None of them was found to be able to automatically load data from a file. See the table of the investigated software applications, and their support of file extensions and tools for loading data in Appendix G.

This pre-study concluded that the problem the researches at CBCS are faced with cannot be addressed with 50 of the most used software applications on the market, and that there is an urgent and real need to develop a new application which can handle this problem.

2.2 Present research

The problem of extracting data together with its implicit schema from tables has been approached in many fields, and different approaches exist. Surveys of systems and methods devoted to the field are found in the following papers: Embley, Hurst, Lopresti and Nagy (2006); Lopresti and Nagy (2000); Silva, Jorge and Torgo (2006); Zanibbi, Blostein and Cordy (2004, 2008).
The problem of interpreting tables is hard and still not solved (Shigarov, 2015). In fact, even human “experts” do not always agree on how to interpret a table (Hu, Nagy, Kashi, Wilfong and Lopresti, 2001), thus we should not expect computers to solve it easily.

2.2.1 The Table understanding problem

In the literature, our problem is known as the table understanding problem, with the subtasks mentioned by Silva et al. (2006) and Shigarov (2015): table location, table segmentation, functional analysis, structural analysis, and lastly interpretation.

Shigarov (2015) mention the following problems for converting tabular information from unstructured to structured form:

- **Table canonicalization** (Douglas, Hurst and Quinn, 1995; Tijerino, Embley, Lonsdale, Ding and Nagy, 2005): How to transform tables and spreadsheets to a so called *canonical form*, which enables for import to relational databases. A *schema* for a canonical table is a finite set of labels $L = \{L_1, \ldots, L_n\}$, and for each label $L_i$ there is a corresponding set $D_i$ called *domain* of $L_i$. Then, for a canonical table $T$, there is a set of functions $T = \{t_1, \ldots, t_m\}$, such that each function maps from a label to its corresponding domain (Tijerino et al., 2005; Embley et al., 2006). Examples of a domain could be row or column values, or a sub-table.

- **Information extraction from tables** (Embley et al., 2006): How to selectively extract facts to populate a given database. This field of research is related to the field of information extraction from text.

- **Table understanding** (Embley et al., 2006): How to recover relations among data values, labels (attributes), and dimensions (domains). This usually includes the following tasks (Hurst, 2001; Silva et al., 2006):

  1. **Table location**: How to differentiate a table from the rest of the document (Tupaj, Shi, Chang and Chang, 1996), for example in HTML documents (Embley, Tao and Liddle, 2005). Typical output is a graphical model with clear outer-boundaries of the table (Silva et al., 2006).

  2. **Table recognition / segmentation**: How to give a physical description of the table, its cells, rows and columns (Silva et al., 2006). See Zanibbi et al. (2004) for a clarifying survey, which breaks it down to the following subtasks: table models, observations, transformations, and interferences. These subtasks interact.

  3. **Functional analysis**: How to classify table areas based on the function it represents in the table. Each cell either holds data or attributes that describe the data (Silva et al., 2006).

  4. **Structural analysis**: How to recover relations between the data cells and the characterising attribute cells, such that the groupings make sense logically (Silva et al., 2006).
5. **Interpretation**: How to interpret and understand a table, in order to extract facts from a table. This involves finding missing information, which can be found in the same document, or by using knowledge from another source or from the real world (Hurst 2000). “A deep interpretation of the table will almost always require context specific knowledge” (Silva et al. 2006, p. 16).

Silva et al. (2006) thoroughly describe the above mentioned steps in extracting information from tables, and present a solution with higher interactions between different steps, i.e. feedback loops. These can reduce errors and contribute to solving previously unsolved problems.

Douglas et al. (1995); Embley et al. (2005); Tijerino et al. (2005) mainly use natural language processing (NLP) to tackle the problem of table understanding, based on domain knowledge about the content. However, Shigarov (2015) argues for that this is not always sufficient in practice, as there are many cases where analysis of spatial and graphical information is also required. Therefore, NLP alone cannot solve the problem.

Shigarov (2015) present a rule engine for functional analysis, structural analysis and interpretation of tables. The information used in these analyses are spatial information, style (typographical) and natural language content of tables. The solution allows for extraction of data from tables presented in Excel files into a database.

Another highly interesting study is Chen and Cafarella (2013). They present a domain-independent spreadsheet extraction system for converting spreadsheet data into relational tuples. There are three components in the system: one that detect the structure of the spreadsheet, one that extracts hierarchical metadata, and one that generates relational tuples.

### 2.2.2 The Semantic web

The extraction of data from tables has been a part of the problem of creating the Semantic Web, i.e. a World Wide Web of data that can be read and processed by computers. Therefore, the field of how to extract data and schema from tables on the Web to open standards such as RDF and OWL formats has been emerging the last couple of decades (O’Connor, Halaschek-Wiener and Musen 2010; Zhao, Zhao and Wang 2010; Han, Finin, Parr, Sachs and Joshi 2008; Yang, Bhowmick and Madria 2005; Tijerino et al. 2005). Syed et al. (2010) present a technique to infer a (partial) semantic model for information in tables and to export the data the tables represent into linked data in RDF-format.

Tijerino et al. (2005) believes in using an ontology as an organizational framework, because of its capability to represent broad classes of entities and relations. The main

---

1. RDF is the abbreviation of Resource Description Framework.
2. OWL is the abbreviation of the Web Ontology Language, and is a family of knowledge representation languages for authoring ontologies.
idea is to manually map to Semantic Web open standards, or using automatic recog-
nition to relate spreadsheet elements to concepts existing in Web knowledge bases
such as DBpedia\footnote{http://dbpedia.org}. This approach could also involve storing the data in databases
with predefined and explicit schemas.

\subsection{2.2.3 Other}

The problem of information extraction from tables has been addressed as part of
other problems. These are a few examples:

- Approach the problem as a part of a database integration problem \citep{ziegler2004}
  \citep{ziegler2004dittrich}
- How, from a Web systems and search engine point of view, to index tabular
data on the web \citep{cafarella2008}
- How to extract tables integrated in HTML to relational schema \citep{embley2005}
  \citep{pivk2007}
- How to transform relational databases to linked data \citep{hu2007}
- How to transform tables to XML documents. \citep{yang2005} present a
  system that transforms flat file data into hierarchical XML data using rule-
based machine learning techniques. The system is implemented in Java and
the effectiveness of the system is demonstrated using real world biological data.
- To find implicit schema in spreadsheets and recognition of construction of
  schemas in spreadsheets used in biodiversity research \citep{bernardo2014}. Difficulties
  of finding this are differences between columns’ order, the label
  used to identify fields and their respective semantics.

\subsection{2.3 Choosing approach}

In this study, we use machine learning in an attempt to make our solution as general
as possible. The following section motivates this choice.

The result from the pre-study motivated us to use new approaches for the problem.
The pre-study of commercial solutions and academic software applications is aimed
to be exhaustive. The result from this study is clear, the problem has not been fully
solved. We hope this study can contribute towards a solution.

In our search for related papers, we found that most prior work focused on the field of
table understanding. This research field tries to solve the problem of understanding
printed tables in pdf files, figures, websites etc. The field seems to have limited
references from other fields we have studied. Most of the progress was made during late 1990’s and early 2000’s.

With this study, we want to go further than the research field of table understanding. We also want to try existing cutting-edge machine learning methods. In spreadsheets, there is large amount of data. We want to make knowledge from this data. To our knowledge, machine learning has rarely been employed on this problem. Chen and Cafarella (2013) is one of these rare examples. We have been inspired by this study.

The research field of machine learning has advanced rapidly the last years, new revolutionary progress is accomplished almost every year. Even better, the progress made in academia is dispersing to the industry. Machine learning is having a huge impact on people’s everyday life.
Chapter 3

Problem Description and Aim

This chapter describes the desired solution for the problem described in Chapter 1 and the aim of this study.

3.1 The desired solution

The case of CBCS could be considered a special case. As concluded from the pre-study, there is significant need for a generic solution. Therefore, the final solution should satisfy a number of criteria.

First, the final solution is required to be generic in terms of the format of the data. Data from other sources, with other structures should be able to be loaded with the same solution. Possible sources includes biomarker discovery and DNA sequencing.

Secondly, the final solution should be able to handle very large datasets, so large that they cannot fit into the working memory of a commodity computer. One solution to this requirement is to divide the dataset into chunks and then analyse each chunk at a time.

Thirdly, the final solution should be able to handle not only text elements, but other types of multimedia objects like images and videos. Today, variety is one of the characteristics of big datasets (Gandomi and Haider 2015). Therefore, to not be able to process these datasets would be to significantly limit the solution’s applicability.

Fourthly, the final solution should be almost fully automatic, with few user interactions during the process. This, in order to be as effective and user-friendly as possible. Without this, the solution will not be distinguished from any of the other solutions because they all need manual templating etc.

Fifthly, the final solution should comply with general best practices for user experience (UX). Therefore, the solution should be deployed as a Software-as-a-Service (SaaS) where the user can access the solution from any device that has Internet
CBCS’s environment requires an intelligent solution where data from any instrument automatically can be learnt and imported. The solution should assign plate maps and controls to individual plates in a graphical interface based on what the user in fact generated. In other words, a solution that learns the structure and format of data files automatically and allows the user to complement it with plate maps. Once imported, pre-formatting of data for the display of various quality measures is envisioned.

3.2 The aim of this study

To build, test, deploy and maintain a complete system described in Section 3.1 is out of the scope of this study. Instead, the aim of this study is to lay down the first few steps towards that goal. We both go broad by considering what the system would look like architecturally, and start a discussion on key technological decisions. We also go deep, by evaluating different approaches for the heart of the system, which is the component that reads, analyses and gives results. We choose to try two approaches.

First, we want to use unsupervised learning to find cells that have the same characteristics. This approach is described in Section 5, the result in Section 6, and the discussion in Section 9.1. The work will contribute to the desired solution by its general applicability.

Secondly, we want to test a hypothesis-driven approach. For this approach, the method is described in Section 7, the result is described in Section 8, and the discussion is found in Section 9.2. This method is influenced by the workflow employed by IBM Watson (Ferrucci, 2012), the question answering computer system that won against humans in the television show Jeopardy! in 2011. The approach will contribute to the long-term solution both short-term, by adapting it to a specific data format, and long-term by generating valuable insights during the building process.

Therefore, the aim of study can be summarised into the following:

1. Investigate if unsupervised machine learning methods can be used to label different types of cells in spreadsheets
2. Investigate if a hypothesis-generating algorithm can be used to label different types of cells in spreadsheets
3. Advise on choices of architecture and technologies for the SaaS solution described in section 3.1

Chapter 4

Theory

4.1 Spreadsheets

In this section, we outline what a spreadsheet is, give context to the specific use case at CBCS, discuss how to interpret them, and finally present the preferred output format.

4.1.1 What is a spreadsheet?

There are many definitions of what a table is, for example by Wang (1996); Embley et al. (2006); Shigarov (2015). An less recent but still up-to-date is given by Cameron (1989): "A table is an object which uses linear visual cues to simultaneously describe logical connections between the discrete content entries in the table. A content entry is the basic component of information in the table".

Many works, including Shigarov (2015), have disregarded context in the rest of the document and only regarded the actual table. In this study we use Cameron (1989) definition of a table, and extend it to hold for spreadsheets, because important pieces of information are given as meta-data above or beneath the table, as seen in Figure 1.3.

A spreadsheet is structured data with an implicit schema (Sved et al., 2010). The schema defines the relations between various data entries and labels.

Let a spreadsheet $S$ be a graphical grid-like structure, with finite number of rows, $r_{\text{max}}$ and columns, $c_{\text{max}}$. The spreadsheet consists of a set of cells, and each cell $e_{i,j}$ is identified by its row $i$ and column $j$.

We define an atomic value as one which is inherently indivisible, i.e. it is more natural to keep it as one value compared to divide it into multiple separate values. Generally, a cell can contain non-atomic values, but in this work we choose to disregard this. Therefore, an element $e_{i,j}$ is equivalent to a cell, and is uniquely
identified by its position $p = (i, j)$. In this study, we will use the terms cell and element interchangeably. Because the number of rows and columns are finite, so is also the number of cells.

Each element holds three attributes, $e_{i,j} = (p, v, t)$, one of them is the mentioned position $p$. The other two are the value $v_{i,j}$ and the datatype $t_{i,j}$. The datatype is derived from the value. In this work, the valid datatypes are string, int and float.

### 4.1.2 How to interpret spreadsheets

There are visual clues on how to interpret the spreadsheet, i.e. how to understand its schema and relations between the elements. Elements on the same row or columns are aligned horizontally and vertically, respectively. These visual clues hold logical meanings of the relations between the elements. Therefore, relative position, $p_{rel} = (i - k, j - l)$, between the two elements, $e_1 = e_{i,j}$ and $e_2 = e_{k,l}$ (where $i \neq k$ and $j \neq l$) is important in interpreting the relation, $r = (e_1, e_2)$, between the two elements. A relation is a statement that holds for the properties of an ordered set of elements. A relation is generally directed, i.e. $r = (e_1, e_2)$ is not equivalent to $r = (e_2, e_1)$.

**Left margin** and **top margin** of tables: Based on the convention of reading tables from the left and from the top, the left margin and top margin of tables are areas where high-precedence domains usually are placed (Douglas et al.; 1995).

Let us define the following relations:

- **block relation**, $r_b = \{e_1, e_2, e_3, \ldots, e_k\}$: An ordered set of elements are in the same block. This is illustrated as the blue square in Figure 4.1. The block relation is uniquely identified by the position of the top left element and its dimensions (number of rows and columns).

- **label-block pair**, $r_{lb} = (e_l, r_b)$: A label element $e_l$ addresses a block relation $r_b$. In Figure 4.1, this is illustrated by the arrow pointing from the green square to the blue square.

- **label-entry pair**, $r_{le} = (e_l, e_e)$: A label element $e_l$ addresses an entry element $e_e$. In Figure 4.1, this is illustrated as the arrow from the green square pointing at the red square.

With these definitions, the terms **entry** and **labels** need to be explained. Our definition is inspired by Wang (1996) and Shigarov (2015). An entry represents data and a label addresses or describes a set of other labels or entries. Specifically for CBCS, a measurement from a microtinter plate is an entry, and annotations about that measurement are labels. “A label can address entries and other labels either in rows or columns only thus labels can form hierarchical relations among themselves. ( . . . ) A label can be a value of a dimension.” Shigarov (2015, p. 931).
4.1.3 Spreadsheets at CBCS

Many of the files CBCS handles have one thing in common: they hold matrices (blocks) with either 96, 384, or 1536 entries, along with other entries and labels. These matrices are usually the central and most important information in the file, they are de facto the measurement output of the machines. These blocks of measurements can be represented in different ways. Figures C.1, C.2, C.3, and C.4 are examples of readout files from the four machines Envision, Microbeta, Victor 2 and Victor 3, respectively. They illustrate the following:

- The matrices can be found multiple times in the same file, as shown in Figures C.1, C.2, and C.3.
- The matrices do not need to be found in the beginning of the file, as shown in Figure C.2.
- The matrices are sometimes headed with numbers and have alphabetical characters on the left side, as shown in Figures C.1 and C.2.
- The whole matrix does not need to be filled, as seen in Figures C.1, C.2, and C.4.

In Figure 4.2, a sketch of a typical input file in HCS is shown. It illustrates two blocks of label-value pairs, or labels. The data block is the only required part, the rest of the parts are optional. The expected size of each part is expressed along with the expected number of characters. Note that in each file, these parts can be repeated more than one time. This sketch is notably similar to what Douglas et al. (1995) presented when analysing tables from the construction industry. Douglas et al. (1995) distinguish between labels and values by using a hierarchy of concepts as the ontological model of the domain. All values associated with a label bear the same relation to this label.
4.1.4 The preferred output format

In the case of CBCS, they prefer a format they can do analyses on. Such a format is shown in Figure 4.3. In this format, each experiment value is on its own row, and is labelled with feature values. In the example shown in the figure, the features are Plate, Repeat, Well, Type, Time, and the name of what the machine is measuring, which is Absorbance @ 405 (1.0s) (A).

4.2 Data schemas and structures

Hadzic, Tan and Dillon (2011), describe different types of data representations, and divide them into relational data, sequential data, semi-structured data, and unstructured data. Semi-structured can also be divided into graph structure and tree structure, where the tree structure is a special case of the graph structure in the case where there is no cycles. The following section outlines what we mean by the term “data structure”, and then follows descriptions of some common data structures.

4.2.1 Definition of data structure

The word “structure” in the Cambridge Dictionary Online (2018) defines the word structure as “[T]he way in which the parts of a system or object are arranged or organized, or a system arranged in this way”. In the spreadsheets we analyse, we assume that there is an unknown set of relationships between the cells, and that there therefore is a system behind how they are arranged and organized. This study is about finding that structure.
Spreadsheets have rows and columns, and therefore could be considered a two-dimensional data structure. Our assumption is that there can be multi-dimensional structures stored in spreadsheets, projected to the two-dimensional space of the spreadsheet. We want to define a structure as something that holds both the data and the schema.

In Section 4.1.1 we defined what an element $e_{i,j}$ is. We here define the data as a set of all elements in the sheet or file:

$$E = \{e_{i,j}\}, \quad (4.1)$$

where $(i, j) \in P \subset \mathbb{N}^2$, $0 \leq i \leq r_{\text{max}}$, $0 \leq j \leq c_{\text{max}}$, and $r_{\text{max}}$ is the largest row index and $c_{\text{max}}$ is the largest column index in the given sheet or file.

In Section 4.1.2 we defined different relations $r$. We here define the schema as all the relations that exist between the elements in the same sheet or file:

$$R = \{r_m : e_{i,j} \rightarrow e_{k,l}, \ m \in T \subset \mathbb{N}\} \quad (4.2)$$

where $i \neq k$ and $j \neq l$ for each $m \in T$. Please note that the special relations mentioned in Section 4.1.2 can be described as a finite set of simple relations.

We now have all the concepts we need to define a structure. We give a structure the following properties:

- A structure consists of a finite set of elements $E$ and a finite set of relations $R$, such that $S = (E, R)$.
- A structure organizes and arranges the elements it holds. This organization and arrangement is defined solely by the relations it holds.
- A structure forms a whole.
- A structure has rules limiting how it can change: The structure can only grow or shrink by a predefined minimum amount. Let us define this as the minimum change block.

### 4.2.2 Relational data model

The relational model is probably the most widely used model for keeping data in a database. The structure is well defined and the schema is known beforehand. All data is represented in terms of tuples (an ordered set of data) grouped into relations (Codd, 2009). Tuples are in practice rows and the relations are tables. The columns in the relation (table) are called attributes.

When the data is organised in a relational model, declarative methods, such as the Sequential Query Language (SQL), can easily and efficiently used to query the data in order to retrieve the desired subset of data (Codd, 1970).
4.2.3 Semi-structured data

Semi-structured data representation is used when the expressive power of the relational model described by Codd (2009) is unable to capture all the relations and semantics in the domain. More complicated data are better represented through trees, lattices, graphs, networks, and other more complex structures. But this also involves higher processing costs (Han, Kamber and Pei 2012, chapter 13). Semi-structured data representations are capable of effectively dealing with two-dimensional relations mixed with other types of relations. The schema or structure of the data and their relations and attributes might not be fixed, and the amount of structure used depends on the purpose. Some examples of sources that Hadzic et al. (2011) mention are XML databases, Resource Description Framework (RDF) databases, molecular databases and graph databases.

The structural organization of a document often indicates the context in which an attribute value occurs and extracting substructures from the original database will keep the context information. Therefore, the attributes of the domain are organized in a hierarchical (tree) or graph structure to enable a more semantic representation of (complex) properties and relations of data objects.

4.2.4 Graph data

Graph data structure is a semi-structured data model consisting of a set of objects and a set of relations between these objects. In this study, the objects are called nodes and the relations are called edges. A graph is directed if the edges have orientation, i.e. the edge $e_1 = \{x, y\}$ is not equal to the edge $e_2 = \{y, x\}$. The opposite case is called an undirected graph. A graph is connected if there is a path between every pair of nodes in the graph, and all nodes are reachable from all the other nodes. The opposite is a disconnected graph (West 2000). A graph is acyclic if there is no set of edges such that one can start from a node, traverse through the edges and come back to the node one started at.

4.2.5 Tree-structured data

The Tree-structured model is a special case of a graph-structured model. In this study it is referred to as a directed and acyclic graph. A special case is a rooted graph, which is a graph in which one of the nodes is distinguished from others, and is called the root. All well-formed trees have only one unique root node. XML data is an example of a rooted well-formed tree (Feng, Dillon, Weigand and Chang 2003). XML files are popular to use when dealing with mining of tree-structured data. Also JSON files follow tree-structured data models.

\(^1\)XML stands for Extensible Markup Language, and defines a set of rules for encoding documents.
4.2.6 Tree vs graph data

There are many efficient algorithms and techniques developed to deal with tree structured data. The complexity of processing tree structures tends to be more manageable than processing graph-structured data. This makes tree mining relatively more attractive to researchers than graph mining (Chi, Muntz, Nijssen and Kok; 2005).

Generally speaking, graphs have many undesirable theoretical properties with respect to algorithmic complexity. One of the common data mining tasks is the systematic enumeration of sub-graphs from a given graph, and to date no efficient algorithm exists to facilitate this task. The inherent complexity in graphs is caused by the existence of cycles (Chi et al.; 2005). However, the number of cycles in real world databases is limited, and the core information can often be represented in a hierarchy. Fortunately, many problems can therefore be modelled using a tree structure without losing too much semantics (Feng et al.; 2003; Chi et al.; 2005). In most cases, the number of cycles in graph instances in a database is limited, or the graphs may even be acyclic, i.e. tree-structured (Chi et al.; 2005).

Many documents found on the web are semi-structured (Wang and Liu; 2000), and XML, JSON, HTML, BibTex, RDF are all examples of standards to express semi-structured data. These documents might not have a fixed schema, but instead might be incomplete or irregular, and is referred to as semi-structured data (Suciu; 2000). Extensive research have been conducted especially on mining XML documents (Tekli; 2016; AliMohammadzadeh, Soltan and Rahgozar; 2006; Feng et al.; 2003; Feng and Dillon; 2004; Wang and Liu; 2000). To conclude, mining XML documents is by large similar to traditional association rule mining, where the problem is to associate XML values with their XML tags.
4.2.7 Sequential data

Sequential data corresponds to an ordered list of objects or events, and is used when the order of the data objects is important. Examples are databases which hold sequences of nucleic acids, sequences of proteins, Website traversal, computer networks, characters in text strings. Sequential pattern mining has emerged as a field to retrieve information from this type of data structures (Mooney and Roddick; 2013).

4.2.8 Unstructured data

Unstructured data has no schema that describes the underlying structure of the data, or the structure is not helpful for the desired processing task. Unstructured data is usually text-heavy. Natural Language Processing (NLP) is one of the more prominent techniques for understanding text data. Other forms of data includes images, audio and video, and this is why mining unstructured data is sometime referred to as multimedia data mining (Zaïane, Han, Li, Chee and Chiang; 1998).

Traditional techniques applied to structured or semi-structured data have limited applications on unstructured data, and the data generally needs major pre-processing and reformatting before starting the actual mining. Therefore, acquiring structured knowledge from unstructured data is often difficult and expensive (Fan, Kalyanpur, Gondek and Ferrucci; 2012).

4.2.9 Big Data

The related term big data has been circulated in the industry and academia during the last decade or so. When mentioned, it is usually vaguely defined. Gandomi and Haider (2015) attempted to define the unique characteristics of big data, and highlighted the fact that the size of the dataset is only one of the characteristics. The other dimensions are velocity and variety, and all three are usually referred to as the three V’s: Volume, velocity and variety (Hsinchun, Chiang and Storey; 2012; Gandomi and Haider; 2015; Kwon, Lee and Shin, 2014). Volume refers to the large quantity of data. Velocity refers to the high speed in which the data is generated or streamed. And variety refers to the various types of data, including text, images, audio and video. 95 percent of these big datasets are unstructured data according to Gandomi and Haider (2015).

4.3 Clustering techniques

Clustering analysis is a sub-field of the unsupervised learning leg of machine learning, where the other leg is supervised learning. Clustering aims to group observations which are considered to be similar into the same group, called cluster, while keeping
large difference between the different clusters. In this section, a number of clustering methods are presented, which are used in this study, after the section on linkage.

4.3.1 Linkage – Similarity measure between sets

Linkage, $D(A, B)$ of set $A$ and set $B$, is the quality of how similar two sets of observations are. Common ways to compute linkage are described below.

**Single linkage**

Single linkage is defined as the minimum distance between an observation in set $A$ and an observation in set $B$ (Florek, Lukaszewicz, Perkal, Steinhaus and Zubrzycki; 1951):

$$D_{single}(A, B) = \min\{d(a, b) : a \in A, b \in B\}. \quad (4.3)$$

**Complete linkage**

Complete linkage, also known as maximum linkage, is usually attributed to Sørensen (1948). It is defined by the maximum distance between the observations in set $A$ and the observations in set $B$:

$$D_{complete}(A, B) = \max\{d(a, b) : a \in A, b \in B\}. \quad (4.4)$$

**Average linkage**

Average linkage, also known as mean linkage, is usually attributed to Sokal and Michener (1958). It is defined as the average distance between all pairs of observations $(a, b)$, $a \in A$, $b \in B$:

$$D_{avg}(A, B) = \frac{1}{|A||B|} \sum_{a \in A} \sum_{b \in B} d(a, b). \quad (4.5)$$

**Weighted linkage**

Weighted linkage is also known as simple average, or WPGMA (Weighted Pair Group Method with Arithmetic Mean). It is similar to average linkage, and is likewise usually attributed to Sokal and Michener (1958). When computing the similarity between a set $C$ and a merge between set $A$ and $B$, it is defined as the arithmetic mean of the similarity between $C$ and $A$ and $C$ and $B$:

$$D_{weighted}(C, (A \cup B)) = \frac{D_{avg}(C, A) + D_{avg}(C, B)}{2}. \quad (4.6)$$
4.3.2 Agglomerative clustering

Agglomerative clustering is a type of hierarchical clustering method which uses a bottom up approach: the algorithm starts by putting each observation in its own cluster, then clusters are successively merged together until there is only one cluster. This produces a dendrogram, i.e. a directed tree of clusters. Here, the leaves represent each observation, and the top node as the whole dataset. The final step of the agglomerative clustering method is to cut the dendrogram at an appropriate level to produce the final clusters. This procedure is described in Algorithm 1 and is implemented by and accessible through Sci-kit Learn’s API (Buitinck, Louppe, Blondel, Pedregosa, Mueller, Grisel, Niculae, Prettenhofer, Gramfort, Grobler, Layton, VanderPlas, Joly, Holt and Varoquaux 2013). In Section 7.3.2, we discuss how we chose where to cut the dendrogram.

\begin{algorithm}
\caption{The algorithm of agglomerative clustering.}
\begin{algorithmic}
\WHILE {The number of clusters is not equal to one}
\STATE Compute the pairwise linkage $D$ between each cluster;
\STATE Merge the two clusters with the smallest linkage between themselves;
\ENDWHILE
\STATE Define the final clustering by cutting the dendrogram at an appropriate level;
\end{algorithmic}
\end{algorithm}

4.3.3 Mean shift clustering

The Mean shift clustering algorithm is employed in the cases where high-density regions are of interest in a two- or multi-dimensional space. In short, the algorithm computes the mean distance of the points within each kernel (cluster) and then shifts the centroid of each kernel to the mean position. Based on the hill-climbing analogy, the algorithm iterates by shifting the position of the centroid to the mean position, until no more points can be added. By then, the cluster has converged to its final position (Comaniciu and Meer 2002).

The algorithm is implemented in the Scikit-learn\footnote{Scikit-learn is an free and open-source machine learning library for the Python language. For more information and API documentation, visit http://scikit-learn.org/ (retrieved 2016-09-20).} Python library, in the clustering module. This implementation uses a flat kernel, the radius of the kernel, called bandwidth, is fixed and given beforehand. In contrast, by using a Gaussian kernel, the mean is calculated using weights that decays with distance exponentially.

4.3.4 Felzenszwalb segmentation method

The Felzenszwalb segmentation method is an image recognition method. A fair question is why an image recognition method used as a clustering method? Well, if
each element is converted to a pixel with a given colour, then an image recognition method could easily be used to cluster the elements.

**Scikit-image** is an open-source image processing library for the python programming language. It implements algorithms and tools and is widely used in research, education, and industry (van der Walt, Schönberger, Nunez-Iglesias, Boulogne, Warner, Yager, Gouillart, Yu and the scikit-image contributors; 2014). The segmentation module in scikit-image consists of a number of segmentation algorithms, including one which implements Felzenszwalb’s efficient graph-based image segmentation algorithm (Felzenszwalb and Huttenlocher; 2004).

The algorithm uses a graph-based representation of the image and performs agglomerative clustering on the pixels. The image is represented as a graph $G = (V, E)$ where each pixel is represented as a node $v_i \in V$, and edges $(v_i, v_j) \in E$ are formed between the pixel and all neighbouring ones. Each edge has an associated weight $w((v_i, v_j))$. Felzenszwalb and Huttenlocher (2004) presented two methods to decide which neighbouring pixels to create edges with. The first option is to use the natural grid of pixels and make edges to all the pixel’s eight neighbours. The weights are then a function of the differences in the intensity or colour of the two pixels. The second option is to use a method to find neighbours. Then, the pixels are mapped to points in a feature space that combines the $(x, y)$ location with the $(r, g, b)$ colour value. Finally, two nodes are considered neighbours if they are close enough in this feature space.

During the agglomerative clustering, evidence of a boundary between two segments are measured using the weights. The algorithm starts with having each node in its own segment. The difference between two segments are measured by considering the minimum weight edge between nodes of the two segments. This minimum weight is compared to the minimum spanning tree edges of the two segments, and the two segments are merged if the weight is smaller than a given threshold. In this sense, each new segment is the minimum spanning tree of its constituent nodes (Felzenszwalb and Huttenlocher; 2004).

Although using this greedy decision of when to merge two segments, the algorithm still yields segmentations that satisfy global properties, especially using the second method for constructing the edges. Another valuable quality of the algorithm is its ability to preserve detail in low-variability regions in the image, and at the same time ignoring detail in high-variability regions (Felzenszwalb and Huttenlocher; 2004).

---

3API documentation and examples are found at [http://scikit-image.org/](http://scikit-image.org/) (retrieved 2016-09-19).


4.3.5  DBSCAN clustering

The DBSCAN clustering method is a method originally proposed by Ester, Kriegel, Sander and Xu (1996). It is implemented and provided through Scikit-LearnDBSCAN. DBSCAN stands for Density-Based Spacial Clustering of Applications with Noise, and it finds high densities in the data and expands clusters from them while discarding outliers as noise. DBSCAN performs well on data containing clusters with similar density (Ester et al. 1996).

4.4  Things to consider when clustering

4.4.1  The Curse of Dimensionality

When Bellmann (1972) first coined the term curse of dimensionality, he referred to the combinational explosion which often occurs when working with large number of dimensions. Klawonn, Höppner and Jayaram (2015) argues that this is usually not the largest problem, as large number of samples tend to have greater negative impact on the computational cost. Nowadays, the term is often used when referring to any type of problem and phenomenon that occurs when working with high-dimensional data.

One of these is the concentration of norm phenomenon. Beyer, Goldstein, Ramakrishnan and Shaft (1999) showed that under a broad set of conditions, distances from any point to its closest and farthest neighbour tend to converge towards each other with increasing dimensionality. Therefore, as the distance usually is a measure of similarity (see Section 4.4.2), it means that all data-points seem to be similar with each other in a high-dimensional space. This phenomenon seems to occur already with as few as 10 or 15 dimensions (Beyer et al., 1999). The $L_k$ norm is often used as a measure of similarity between samples. Aggarwal, Hinneburg and Keim (2001) shows that this problem is sensitive to the value of $k$ when using the $L_k$ norm as a measure of distance, and propose the fractional norm $L_{\frac{1}{2}}$ as one of the better choices.

Durrant and Kaban (2009) provide a converse discussion on when nearest neighbour actually is meaningful in high-dimensional spaces.

Another one is the hubness phenomenon. Often in high-dimensional data, a few data points – referred to as the hubs – occur significantly more frequently among the $k$ nearest neighbours of other data points (Klawonn et al. 2015). This is notable, as one would expect that each point would roughly occur among the $k$ nearest neighbours equally often, especially for uniformly distributed data. Radovanović, Nanopoulos and Ivanović (2010) coined the phenomenon, and showed that it is an inherent property of data distributions in a high-dimensional vector space.

---

Hub points do not cluster well because they have low inter-cluster distance. It is hard to decide which cluster to put them in because they are close to many points in many clusters. Radovanović et al. (2010) also coin the term antihub, a point which is almost never the nearest neighbour to other points. Points that are significantly dissimilar to all other points are usually referred to as outliers. Outliers occur because of many reasons, for example due to measurement errors. They tend to be a problem when clustering, and are not seldom identified and removed before the actual analysis. This is problematic because Radovanović et al. (2010) suggest that high dimensionality induces antihubs. These antihubs may not be considered to be outliers in lower dimensions, but may appear to be stronger outliers in high dimensions.

How does these phenomena effect cluster analysis? Klawonn et al. (2015) concludes that “[c]lustering high-dimensional data is a difficult task”. We can note that high-dimensional clustering seem to be much more than just choosing the most common clustering method and the most common distance metric.

4.4.2 The choice of similarity measure

There are many different ways to define similarity between two observations. The choice of similarity (or distance) measure can have a large influence on the result of the clustering algorithm. When working with numeric values in the real space, a measure of distance is commonly used, for example Euclidean distance, Manhattan distance, Cosine distance, Mahalanobis distance, etc.

4.4.3 Dimensionality reduction techniques

Dimensionality reduction techniques are often used to address the problem of the curse of dimensionality, and to make it easier to visualise the data. PCA and SVD are two commonly used techniques.

Singular Value Decomposition

Singular Value Decomposition (SVD) is a well-known method in the field of linear algebra to factorize matrices into compositions, see for example Halko, Martinsson and Tropp (2009a); Allaire, Trabelsi and Kaber (2008) or Benner, Bollhöfer, Kressner, Mehl and Stykel (2015) for reference and Stewart (1993) to read about the early history of SVD. Every rectangular matrix has a singular value decomposition. In the contrary, the eigendecomposition – another commonly used method to factorize matrices – can only be applied to diagonalizable matrices. A useful application of SVD is to compute the truncated SVD, which is a lower-ranked approximation of a matrix. But let us first define the SVD.

7In fact, SVD is a generalization of eigendecomposition.
The singular value decomposition of the matrix $A$ of size $m \times n$, $n \leq m$, is given by

$$A = U\Sigma V^T. \quad (4.7)$$

Here, $U$ is a unitary matrix of size $m \times m$, $\Sigma$ is a diagonal matrix of size $m \times n$ and $V^T$ is the transpose of a unitary matrix $V$ of size $n \times n$. The diagonal entries of the matrix $\Sigma$ are called singular values. The columns of $U$ are called the left-singular vectors of $A$, and the rows of $V$ are the right-singular vectors of $A$. The rank of $A$ is equal to the number of non-zero singular values of $A$.

Let matrix $A$ have rank $r$. Then, $A$ can be approximated by another matrix $\tilde{A}$ with lower rank $\tilde{r}$, $\tilde{r} < r$, where $\tilde{A}$ is given by

$$\tilde{A} = \sum_{i=1}^{\tilde{r}} v_i \sigma_i v_i^T. \quad (4.8)$$

Here, $\sigma_i$ is the $i$th singular value and $V_i$ is the corresponding singular vector in $\sigma$ and $V$. $\tilde{A}$ is notably the best approximation of $A$ compared to all other matrices of rank $t$ (Allaire et al. 2008).

In practice, it is computationally costly to calculate the SVD. Therefore, often only a subset $t$ of the singular values, the left- and right-singular vectors are computed, and the rest are discarded. This is called the truncated SVD.

**Principal Component Analysis**

Principal Component Analysis (PCA) is one of the more prominent and commonly used methods for dimensionality reduction (Tipping and Bishop 1999).

The performance of PCA is assessed through analysis of the percent of variance in the dataset that is retained while reducing the number of dimensions. Retaining thresholds of 99 percent, 95 percent, or 90 percent are commonly used, depending on the problem at hand.

**4.4.4 Clustering evaluation**

What makes a clustering analysis a good one? Generally, there are two types of measurements used to evaluate clustering analyses (Rendón, Abundez, Arizmendi and Quiroz 2011). In the literature, they are usually referred to as indices:

- **An external index** measures how well the clustering matches externally supplied pieces of information such as class labels. Frequent measures are entropy and purity.

- **An internal index** measures the clustering structure without respect to external information, and instead only considers information intrinsic to the data alone. The Silhouette coefficient, the Calinski-Harabaz index and the Davies-Bouldin index are examples of internal indices.
In this study, the three following internal indices are chosen to evaluate our clustering analysis: the silhouette coefficient, Calinski-Harabaz index, and Davies-Bouldin index. Internal indices are chosen because they can be used without external information. These specific indices are chosen because they seem to be the only ones implemented in Scikit-Learn as of 2018-11-26. Because Scikit-Learn is a trusted API used by many in the industry, these three indices seem to be a good start (Who is using scikit-learn? [2018]).

All three indices estimate the within-cluster similarity, which is how coherent the points in each cluster are, and the dissimilarity between clusters, that is how different points belonging to one cluster are to points outside of its own cluster.

- **Silhouette coefficient**: The index is bounded to $[0, 1]$ where bigger is better. Values near 0 indicate overlapping clusters.

- **Calinski-Harabaz index**: The index is defined as ratio between the within-cluster dispersion and the between-cluster dispersion. The index is not bounded. The higher the index the better the separation is. It is also supposedly relatively fast to compute.

- **Davies-Bouldin index**: The index is defined as the ratio of within-cluster distances to between-cluster distances. It measures the distance between clusters’ centroids. It is restricted to using the Euclidean distance function. Lower index is better.
Chapter 5

Method – The Unsupervised Learning Approach

This chapter outlines the method for the unsupervised learning approach.

5.1 Pre-processing

This section describes how a set of input files are read and transformed into a $r \times c$ matrix $M$, where each row represents a sample and each column represents a feature. In this case, each cell in the input files are seen as a sample. The python libraries Pandas\(^1\) and NumPy\(^2\) are used to parse the Excel or text files.

5.1.1 The intermediate matrix and primitive variables

First, an intermediate matrix $T$ is formed. In $T$, each cell is represented by a row. Empty cells are excluded. A cell containing a string of multiple words is preliminary not split into multiple samples, one word in each sample, but is instead seen as a single sample. Thus, the complete string is represented as a single row.

The following were chosen to be the features of each sample, and thus constitutes the columns in the $T$ matrix:

- **column**: The index of the column in the spreadsheet where the cell was found (starting from column 0).
- **row**: The index of the row in the spreadsheet where the cell was found (starting from 0).

\(^1\)Read more at: [https://pandas.pydata.org/](https://pandas.pydata.org/) (retrieved 2018-10-15).
• **datatype**: What datatype the value in the cell has, for example float, integer, string.

• **value**: The literal value of the cell as given in the input file. All values are stored as strings.

• **filename**: The name of the file where the cell was found.

• **sheetname**: The name of the sheet where the cell was found. If the cell was found in a csv or similar format, then the value is empty.

Other features could have been chosen to be included in this list; for example attributes found in cells in Excel spreadsheets such as colouring, font size, border around the cell etc., or features describing the surrounding of the cell, or the cell relative to other cells. But to only include the most basic, primary, and non-derived features, none of these were included.

\( T \) is a feature matrix consisting of samples as rows and features as columns. Now we want to use a machine learning method on our feature matrix. However, machine learning methods cannot handle text values as input, so the columns in \( T \) which hold text values need to be transformed into numerical values. One way of doing this is by using one hot encoding.

### 5.1.2 One Hot Encoding of categorical features

The columns **datatype**, **filename**, and **sheetname** in the \( T \) matrix are all categorical variables, i.e. there is no intrinsic order within each of them. The one hot encoding is a process by which a categorical variable is transformed into several variables, one variable for each unique value in the original variable. Practically, this means that the column representing datatypes is replaced by the columns **datatype** float, **datatype** integer and **datatype** string. At rows that before the transformation had the value float in the column **datatype** will after the transformation have the value 1 in the **datatype** float column, and 0 in the rest of the new columns.

We do multiple one hot encodings of the matrix \( T \), and call the resulting matrix \( M \).

The obvious change when using one hot encoding is that it significantly increases the dimensionality of the feature space. Before the encoding, the matrix \( T \) has 6 dimensions. After the encoding, the resulting matrix \( M \) has

\[
\begin{align*}
\text{number of features} & = 2 + d_{dt} + d_f + d_s \\
& \quad \text{(5.1)}
\end{align*}
\]

columns. Here, the number two comes from the two features row and column, \( d_{dt} \) is the number of unique datatypes, \( d_f \) is the number of unique filenames, and \( d_s \) is the number of unique sheetnames. This matrix is sparse; a vast majority of the values in the columns formed during the one hot encoding are 0.

An example: We read 15 files, all together they contain 63 sheets. This results in a feature matrix with 131,944 rows (cells) and 12,256 columns (feature dimensions).
The density of the matrix, that is the fraction of non-zero elements, is less than $4 \times 10^{-4}$.

The dataset resulting in only 15 input files is large, which makes it hard to work with. For example, to compute the rank of the matrix takes more than an hour on a commodity laptop (A MacBook Pro with a 2.7 GHz Intel core i5 processor, and 16 GB RAM).

5.2 The input data

5.2.1 Assumptions on input files

The following assumptions are made on the input files for both approaches:

- The file is text based.
- The file is finite and fits into the working memory of a commodity computer.
- A text field is considered to be atomic, i.e. a cell only contain one value.
- Data type is identified by the content of the text field. A limited number of datatypes are found in the file, namely
  - strings,
  - integers,
  - floats.
- The delimiter of the text fields is unknown.
- The same delimiter is used throughout the entire file.
- Each text field is a piece of data and is inherently valuable. It should therefore, if possible, be preserved throughout the process until the user actively discards it.

5.3 Dataset A

The input dataset A is put together to make it easy for the clustering algorithm. Therefore, only files coming from the Victor 3 machine were included. Table 5.1 shows the name of the files and their respective number of sheets. Many of the files contain similar sheets, Table 5.2 illustrates this fact. Table 5.3 gives an overview of other basic facts about the dataset. For example, on average there are 6.1 columns in each sheet, and 337 rows.
### Table 5.1: The name of the 15 files in dataset A, together with their respective number of sheets.

<table>
<thead>
<tr>
<th>Filename</th>
<th>number of sheets</th>
</tr>
</thead>
<tbody>
<tr>
<td>Abs 405 nm kinetics 384020000371345120160315.xls</td>
<td>5</td>
</tr>
<tr>
<td>Abs 405 nm kinetics020000321198120150330.xls</td>
<td>5</td>
</tr>
<tr>
<td>Abs 405 nm kinetics020000321268320151023.xls</td>
<td>4</td>
</tr>
<tr>
<td>Abs 405 nm kinetics020000321297420151202.xls</td>
<td>5</td>
</tr>
<tr>
<td>Abs 405 nm0200001710682320140529.xls</td>
<td>4</td>
</tr>
<tr>
<td>Abs 405 nm0200001710692420140529.xls</td>
<td>4</td>
</tr>
<tr>
<td>Abs 405 nm0200001711723220141216.xls</td>
<td>4</td>
</tr>
<tr>
<td>Abs 405 nm0200001711723320141216.xls</td>
<td>4</td>
</tr>
<tr>
<td>Abs 405 nm0200001711733620141216.xls</td>
<td>4</td>
</tr>
<tr>
<td>Abs 405 nm0200001711863620141216.xls</td>
<td>4</td>
</tr>
<tr>
<td>Abs 405 nm0200001711863720150302.xls</td>
<td>4</td>
</tr>
<tr>
<td>Abs 405 nm0200001711873820150303.xls</td>
<td>4</td>
</tr>
<tr>
<td>Abs 405 nm0200001711883920150303.xls</td>
<td>4</td>
</tr>
<tr>
<td>Abs 405 nm0200001711904120150303.xls</td>
<td>4</td>
</tr>
</tbody>
</table>

In Figure 5.1 the frequency of features is plotted. The name of sheets and name of datatypes are features many order of magnitudes larger in frequency compared to the rest of the features. The frequency of these features are not interesting at this stage. The least frequent features, with frequency of 1 up to 3 or 5, are those representing the right tail of the frequency spectrum. This tail could be long, as the number of features usually range in thousands or tens of thousands. In Figure 5.2 the features’ frequency is plotted, with the most frequent and the least features left out.

### 5.4 Dimensionality reduction

When reading our dataset, our feature matrix $M$ has a high number of dimensions, usually thousands. The feature matrix $M$ is so large that the time to process it is limits the analysis. To illustrate with an example: in the dataset A, 15 files are read, containing in total 63 sheets. This results in a matrix of 100,754 data points and 12,255 features. The data is clustered using one of the few practically plausible methods available – the mini-batch K Means. When computing the silhouette score of the clustering result, it did not finish in 56 minutes.

In Section 4.4.3 the theory of SVD and PCA is discussed. Sci-kit Learn’s implementations of these two methods are chosen to test to reduce the dimensionality of the input data. According to the documentation of the API, truncated SVD

---

Table 5.2: Name of sheets and their total frequency in the dataset.

<table>
<thead>
<tr>
<th>Sheetname</th>
<th>Frequency</th>
</tr>
</thead>
<tbody>
<tr>
<td>Errors</td>
<td>3</td>
</tr>
<tr>
<td>List ; Plates 1 - 1</td>
<td>9</td>
</tr>
<tr>
<td>List ; Plates 1 - 12</td>
<td>1</td>
</tr>
<tr>
<td>List ; Plates 1 - 2</td>
<td>1</td>
</tr>
<tr>
<td>List ; Plates 1 - 3</td>
<td>1</td>
</tr>
<tr>
<td>List ; Plates 1 - 46</td>
<td>1</td>
</tr>
<tr>
<td>List ; Plates 1 - 9</td>
<td>1</td>
</tr>
<tr>
<td>List ; Plates 1 - 99</td>
<td>1</td>
</tr>
<tr>
<td>Notes</td>
<td>15</td>
</tr>
<tr>
<td>Plate</td>
<td>15</td>
</tr>
<tr>
<td>Protocol</td>
<td>15</td>
</tr>
</tbody>
</table>

Table 5.3: Statistics over the input dataset called dataset A. The metrics are the mean, the standard deviation, minimum, median, and maximum values. These are applied to rows per sheet, columns per sheet, cells per sheet, and sheets per file.

<table>
<thead>
<tr>
<th></th>
<th>Rows per sheet</th>
<th>Columns per sheet</th>
<th>Cells per sheet</th>
<th>Sheets per file</th>
</tr>
</thead>
<tbody>
<tr>
<td>mean</td>
<td>337.0</td>
<td>6.1</td>
<td>1599.3</td>
<td>4.2</td>
</tr>
<tr>
<td>std</td>
<td>1349.7</td>
<td>2.0</td>
<td>7360.9</td>
<td>0.4</td>
</tr>
<tr>
<td>min</td>
<td>3</td>
<td>1</td>
<td>3</td>
<td>4</td>
</tr>
<tr>
<td>median</td>
<td>31</td>
<td>6</td>
<td>31</td>
<td>4</td>
</tr>
<tr>
<td>max</td>
<td>9505</td>
<td>12</td>
<td>57030</td>
<td>5</td>
</tr>
</tbody>
</table>

is a method specifically suited for sparse data. Truncated SVD operates directly on sample vectors, instead of on a covariance matrix (Buitinck et al., 2013). Our dataset is notably sparse, and the method is therefore arguably worth evaluating in addition to the popularly applied PCA method.

When using Scikit-Learn’s implementations of truncated SVD and PCA, one can choose which eigenvalue solver the method should use. In this study we choose to compare the default randomized solver against the ARPACK solver. The randomized solver was framework proposed by Halko, Martinsson and Tropp (2009b) to construct randomized algorithms that compute partial matrix decompositions. ARPACK is a framework written in Fortran by Lehoucq, Sorensen and Yang (1998).

5.5 Choosing learning model

The reason we chose unsupervised learning is because it would require manual work to mark up our data with labels. We chose to try unsupervised learning first, to see if it is useful.

Because our dataset is both high-dimensional and large sample-wise, some clustering (retrieved 2018-10-20).
methods are more efficient than others. A method is required that scales well with both dimensions and number of samples used. Sculley (2010) suggests using mini-batch optimization for k-means clustering to address the extreme requirements for latency, scalability, and sparsity encountered in web applications.

Unfortunately, when using Scikit-Learn’s implementation of mini-batch k-means clustering, one cannot choose which distance metric to use. The Euclidean distance is the only option. This is the same for Scikit-Learn’s implementation of the scoring functions to use when evaluating the clustering (Calinski-Harabaz Index and Davies-Bouldin Index, introduced in Section 5.6).

5.6 Evaluation

Internal indices are used because then we do not need to use external information, for example by marking up our dataset with labels. Choosing the internal index with which to evaluate is hard. Kovács, Legány and Babos (2005) gives an example of that.
Sci-kit Learn has implementations of three internal indices. When testing these scores, the Silhouette score has shown to have performance issues on our dataset, and is therefore excluded from the study. When scoring clusterings of a dataset of 20 000 points and 12 255 feature columns, it took hours to compute the silhouette coefficient, compared to minutes to fit the mini-batch K-means clustering and seconds to compute Calinski-Harabaz Index and Davies-Bouldin Index.

In addition to the three indices already mentioned, we implemented our own index. Unfortunately it turned out to be computationally too expensive to be used. It is based on Matthews correlation coefficient for binary classification. A matrix of MCC:s is computed, one index for each string and cluster. For each string cell and each cluster, it regards two true labels and two predicted labels: the two true labels are the string and everything else. The two predicted labels are the given cluster and everything else. Then it computes the Matthew’s correlation coefficient based on these.

We also want to make sure our results are independent on our choices of hyper-parameters. Therefore, we need to be able to search through different choices and evaluate the results based on our objective criteria, which are the internal indices. The hyper-parameters include the number of clusters and other parameters that goes into the clustering method.

### 5.6.1 Visualising the results

The clustering is done in a high-dimensional space, and there are different ways to visualise the results. In this study we can go back to our original form of the data, i.e. spreadsheets, and visualise those. For the unsupervised learning, we have chosen to only visualise one cluster at the time, if nothing else is mentioned. It would be difficult to distinguish the clusters from one another when the number of clusters is around 100.

---

User guide:

Chapter 6

Result – The Unsupervised Learning Approach

In this chapter, the results of the unsupervised learning approach is presented.

In section 6.1, dimensionality reduction is evaluated as a pre-processing step before the actual clustering. In section 6.2, clustering with different parameters are compared on a dataset containing a broad set of data formats. In section 6.3, we make it easier for the clustering method by narrowing the data format down by only considering one file format.

6.1 Dimensionality reduction

We evaluate this because of the problems that might be seen when working with high-dimensional data. The term curse of dimensionality is the umbrella term for these problems, as discussed in Section 4.4.1.

6.1.1 Comparing Dimensionality reduction methods

As discussed in Section 4.4.3, PCA and SVD are methods for reducing dimensionality. To evaluate these methods, we use the ratio – the sum of variance in the resulting dimensionality reduced dataset divided by the sum of variance in the original dataset. A higher ratio for a given dimension is better than a lower ratio.

Figure 6.1 shows four different methods of dimensionality reduction, namely Truncated SVD with a randomized solver, Truncated SVD with ARPACK solver, PCA with randomized solver, and PCA with randomized solver. All four methods have roughly the same performance. All of them reach a cumulative variance of 90 percent at around 90 components. Around 75 percent of the variance is reached using 8 components. When adding a new component after the 10th, the contribution of variance explained is notably low.
Figure 6.1: Cumulative sum of variance ratio using four different methods of dimensionality reduction, namely Truncated SVD with a randomized solver, Truncated SVD with ARPACK solver, PCA with randomized solver, and PCA with randomized solver. In the embedded figure the first 20 components are shown, together with threshold levels of 75 percent and 95 percent of cumulative variance retained after the reduction. None of the four methods reach the level of 95 percent within the first 200 components.

Figure 6.1 shows that both the PCA methods slightly outperform truncated SVD during the first 10 components. When evaluated for more than 10 components, the four methods’ performance is almost the same throughout and until 200 components. This implies that neither the choice of PCA vs SVD nor the solver (randomized vs ARPACK) have significant impact on performance when applied to our dataset.

6.1.2 The rank of the feature matrix

In this section we approach the same problem from the other end. Here, we compute the rank of the feature matrix to see how many of the column feature vectors that are linearly dependent, if any.

The rank is computed using NumPy’s method matrix_rank. In our dataset, the matrix m has 1.6 billion elements \((131,944 \times 12,256)\). This means that the largest possible rank is \(r = \min(r_N, c_N) = 12,256\) (where \(r_N\) is number of rows and \(c_N\) is number of columns). The rank took more than an hour to compute, and resulted in rank 12,254, which is merely 2 less than the maximum rank. This means only two columns out of 12,256 are linearly dependent on the other ones. This was expected, since all of the columns created by the one-hot encoder are perpendicular to each other.

Both the results from the dimensionality reduction methods in Section 6.1.1 and from the rank analysis above imply that the vast majority of the dimensions in dataset A are needed to describe our data in an acceptable way. Therefore, none of the following clustering analyses are based on dimensionality reduced data, and all the 12,256 dimensions are used.
6.2 Clustering comparison

The following section presents the findings from the clustering of our dataset. It is clustered using mini-batch K-means, while varying the parameters number of clusters and batch size. The two internal indices Calinski-Harabaz and Davies-Bouldin are used to compare results of the clusterings. In order to compare the Calinski-Harabaz index with the Davies-Bouldin index, they are normalized to the range $[0, 1]$ in this study, where 0 is the worst and 1 is the best.

Table 6.1: Heatmap of unit-normalized Calinski-Harabaz index, with batch size on x-axis and number of clusters on y-axis.

<table>
<thead>
<tr>
<th># of clusters</th>
<th>70</th>
<th>100</th>
<th>110</th>
<th>150</th>
<th>2000</th>
<th>3000</th>
<th>4000</th>
<th>5000</th>
<th>6000</th>
</tr>
</thead>
<tbody>
<tr>
<td>70</td>
<td>0.88</td>
<td>0.81</td>
<td>0.77</td>
<td>0.87</td>
<td>0.88</td>
<td>1.00</td>
<td>0.97</td>
<td>0.99</td>
<td>0.97</td>
</tr>
<tr>
<td>80</td>
<td>0.64</td>
<td>0.65</td>
<td>0.62</td>
<td>0.65</td>
<td>0.74</td>
<td>0.79</td>
<td>0.80</td>
<td>0.80</td>
<td>0.79</td>
</tr>
<tr>
<td>90</td>
<td>0.40</td>
<td>0.53</td>
<td>0.51</td>
<td>0.54</td>
<td>0.57</td>
<td>0.66</td>
<td>0.65</td>
<td>0.63</td>
<td>0.64</td>
</tr>
<tr>
<td>100</td>
<td>0.30</td>
<td>0.35</td>
<td>0.36</td>
<td>0.45</td>
<td>0.46</td>
<td>0.53</td>
<td>0.52</td>
<td>0.52</td>
<td>0.53</td>
</tr>
<tr>
<td>110</td>
<td>0.29</td>
<td>0.25</td>
<td>0.27</td>
<td>0.32</td>
<td>0.38</td>
<td>0.41</td>
<td>0.40</td>
<td>0.42</td>
<td>0.43</td>
</tr>
<tr>
<td>120</td>
<td>0.11</td>
<td>0.18</td>
<td>0.21</td>
<td>0.27</td>
<td>0.27</td>
<td>0.34</td>
<td>0.33</td>
<td>0.34</td>
<td>0.33</td>
</tr>
<tr>
<td>130</td>
<td>0.09</td>
<td>0.09</td>
<td>0.14</td>
<td>0.14</td>
<td>0.25</td>
<td>0.25</td>
<td>0.27</td>
<td>0.28</td>
<td>0.27</td>
</tr>
<tr>
<td>140</td>
<td>0.01</td>
<td>0.03</td>
<td>0.08</td>
<td>0.09</td>
<td>0.13</td>
<td>0.19</td>
<td>0.19</td>
<td>0.19</td>
<td>0.21</td>
</tr>
<tr>
<td>150</td>
<td>0.00</td>
<td>0.02</td>
<td>0.01</td>
<td>0.01</td>
<td>0.07</td>
<td>0.17</td>
<td>0.15</td>
<td>0.14</td>
<td>0.14</td>
</tr>
</tbody>
</table>

Because we primarily are interested in the relationships of the cells in sheets called Plate in files of type Victor 3, we will only consider the results for these sheets if nothing else is specified. Because some of these sheets have a large number of rows, it is difficult to visualise all of them in one page. Therefore, where it is needed, the sheets are truncated when presenting results.

What we want to achieve with the clustering is to have one string in each cluster. For example, cells containing barcode is clustered together, cells containing the string repeat are clustered together, etc. No clusterings give satisfying results, regardless of their scoring on the Calinski-Harabaz and Davies-Bouldin indices. The Calinski-Harabaz index and the Davies-Bouldin indices weakly covary with each other, with a covariance of 0.03. They suggest different candidates of clusterings to be the best.

According to the Calinski-Harabaz index presented in Figure 6.1, the best clustering is given by choosing batch size to 20,000 and cluster size to 70. But when looking at the results in all sheets called Plate, there is no real pattern a human would consider. Either the cluster is spread out throughout the sheet as in Figure 6.2 or the entire sheet is in the same cluster, as in Figure 6.3.

According to the Davies-Bouldin index, given in Figure 6.2, the best and second best clustering are the ones with the following parameters: batch size = 60,000 and number of clusters set to 70, and batch size = 20,000 and number of clusters = 80. But these choices of parameters yields trivial results, or poor results. When looking at sheets called Plate, all cells are clustered into the same cluster, please see for example Figure E.1 in Appendix E. Figure E.2 is an example of poor clustering, with batch size 20,000 and number of clusters 80.
Figure 6.2: Sheet Plate in file Abs.405_nmm0200001711904120150303.xls. Cells belonging to cluster $C_i, i = 17$, are highlighted in green.

Table 6.2: Heatmap of unit-normalized Davies-Bouldin index, with batch size on x-axis and number of clusters on y-axis.

It seems like all clusterings give the same poor result, regardless of how good the score on the two indices is.
Based on the experiments described in Section 6.2, the clustering does not provide any satisfying result, regardless of the settings of parameters batch size and number of clusters. Therefore, experiments on different datasets are done and described in this section.

6.3.1 Same files as before

The following describes an experiment undertaken on dataset A, which is described in Section 5.3. For this experiment, the same files are read, but only the sheets called Plate is considered. Because there is one sheet called Plate in each file, it will be the same number of sheets. This gives a dataset of 16,302 data points, representing
cells only coming from sheets called Plate. The number of features (dimensions) is 207, which is significantly lower compared to the 12,256 features given when looking at all sheets.

When reducing the dimensionality of this dataset, around 95 percent of the variance is propagated using the first 18 components, see Figure 6.4. This is significantly fewer components to describe the same variance compared to the analysis in Section 6.1.1. All the dimensionality reduction methods have approximately the same performance.

This dataset still produces poor clustering, regardless of the choice of clustering parameters (batch size and number of clusters). Some clusters captures the entire sheet, other clusters are scattered over the sheet.

6.3.2 Increased number of files

Maybe there are too few data points to produce good clustering results? Therefore, a new experiment is done where 75 files of type Victor 3 are read, see Tables C.3 and C.4 for a full list of files. For reference, we call this the dataset B. Also for this experiment, only sheets named Plate are considered. One sheet per file sums up to 75 sheets in total. When this is read, a feature matrix of 63,285 data points and 542 features is generated.

Once again, the dimensionality reduction gives satisfying results. Around 95 percent of the variance can be explained by the first 40 components, please see Figure 6.5. Just as before, all the dimensionality reduction methods have approximately the same performance.

When clustering, it consistently gives poor results, regardless of reducing or not
reducing the dimensions before, and regardless of the choice of batch size and number of clusters.

**Figure 6.5:** Dimensionality reduction on dataset B, including only sheets called Plate. Cumulative sum of variance ratio between after and before the dimensionality reduction. This is done for four different methods, namely Truncated SVD with a randomized solver, Truncated SVD with ARPACK solver, PCA with randomized solver, and PCA with randomized solver.
Chapter 7

Method – The Hypothesis-generating Approach

In this chapter, the proposed algorithm is described. It is divided into modular components with clear interfaces. This makes it easy to modify the algorithm by updating specific components without affecting the rest of the algorithm. Therefore, each component has input and output variables comparable with required respectively provided interfaces in the Uniform Modeling Language (UML) terminology.\footnote{Read more about UML at \url{http://www.uml.org/} [2016-09-28].} Pre-processing, hypothesis generation, hypothesis evaluation, hypothesis selection, and lastly learning are the sequential modules. Please see Table 7.1 for the interface of each module.

7.1 Specialisation

In order to contribute short-term to the desired solution described in Section 3.1, the algorithm is adapted to the Victor 2 and Victor 3 file types, and only the sheet called Plate. Furthermore, it only accepts one sheet at the time being. In other words, when built-in rules and architectural choices are done, they are made with a sheet Plate in a Victor 2 or Victor 3 file in mind. In this way, the algorithm is specialised in the data format found in these sheets.

7.2 Pre-processing

In this module of the algorithm, a raw input file is read and turned into a python dictionary of elements (for definition of an element, see Section 4.1.1). In the first version of this algorithm, csv, xls, and txt file formats can be handled.

As an attempt to make the algorithm be able to handle a broader range of file
Table 7.1: To divide the solution program into separate components, the data loading pipeline is divided into a sequence of components with defined interfaces.

<table>
<thead>
<tr>
<th>Step</th>
<th>Input variables</th>
<th>Output variables</th>
</tr>
</thead>
<tbody>
<tr>
<td>1. Reading and preprocessing: <em>The input file is read and preprocessed.</em></td>
<td>- An input file from user</td>
<td>- A dictionary of elements</td>
</tr>
<tr>
<td>2. Hypothesis generation: <em>Hypotheses are generated using different methods and approaches.</em></td>
<td>- Dictionary of elements</td>
<td>- Labeled nd array</td>
</tr>
<tr>
<td>3. Hypothesis evaluation: <em>An objective function is used to score each hypothesis.</em></td>
<td>- Labeled nd array - Dictionary of elements - Objective function - Weights</td>
<td>- A scalar representing the scoring of the hypothesis</td>
</tr>
<tr>
<td>4. Hypothesis selection: <em>User chooses a preferred hypothesis from a ranked list of hypotheses.</em></td>
<td>- List of labeled nd arrays - List of scoring scalars - User interaction</td>
<td>- The selected labeled nd array</td>
</tr>
<tr>
<td>5. Learning: <em>Update of weights.</em></td>
<td>- List of labeled nd arrays - A selected labeled nd array</td>
<td>- Weights</td>
</tr>
</tbody>
</table>

formats, a frequency analysis on the characters in the file is implemented. The first version of this module counts the characters like comma, space, tab, full stop, etc. The most frequent of these characters is taken as the delimiter. Hence, the user is not required to manually adapt his or her input file to use the correct delimiter before importing a file.

A pandas DataFrame\(^2\) is used as an intermediate format of the data. Elements containing empty strings, single dashes, single space, and alike are not considered to add more noise than information. They are therefore removed from the element dictionary.

7.3 Hypothesis generation

In this module, the goal is to generate hypotheses of how to interpret the text file. A large number of hypotheses is preferred because the more hypotheses we have, the larger the chance is that one of them is, or is close to, the correct interpretation.

Figure 7.1 presents the conceptual workflow of the algorithm. The algorithm starts with a single hypothesis $H_1$. For each decision the algorithm confronts, new hypotheses are generated – one for each option to the given decision.

Let a hypothesis be defined by two data objects, a labeltree and a Nordron N-Dimensional array, henceforth called nd array or just nd. A labeltree is defined as a tree structure where each node has a value, a parent node and any number of children nodes. In this way, it orders and arranges the relations between different dimensions. An nd is an n-dimensional array stored as a dictionary where the keys are tuples. In this way, the array can be extremely sparse and therefore a large number of dimensions can be used.

![Figure 7.1: Decisions are represented by diamond squares and hypotheses are represented by \( \mathcal{H} \) symbols.](image)

The hypothesis generation step can be described by two sub-steps. First, it clusters the elements to find blocks, then it extracts relations between these blocks and other elements. Ideally, for every choice the algorithm does, there should be multiple hypotheses covering the available options. For this first version of the algorithm, the main choices are which clustering method to employ, and subsequently choosing parameters for the clustering.

The complete set of decisions and options implemented is presented in Figure 7.2. The first decision is the choice of clustering method. For the agglomerative clustering method, the second and third decisions are the choice of linkage, and the choice of cut. The theory of linkage is described in Section 4.3.1. The following definitions of linkage are implemented in the algorithm: average, complete, single, weighted.
7.3.1 Clustering

To find blocks, the clustering methods use data on position and datatype of each element. Prior to the clustering, the datatypes were mapped to a number ranging from 0 to 1. This was done in order to make use of machine learning frameworks such as Scikit-learn and SciPy. These two frameworks only use numbers as input to their machine learning methods. One way of seeing this mapping is as an image, where each pixel represents a cell in the spreadsheet. The colour of each pixel is decided by the datatype of the cell it represents.

The choice of which value to give to each datatype is not trivial. The values were chosen based on the programmatic relation the datatypes have in the Python language. Firstly, an integer can be casted to a float without loosing information, but a float cannot be an integer without loosing information. Secondly, a float can be casted to a string without loosing information, but not the other way around without loosing information. By this, integers is a subset of floats, and floats is a subset of strings. See Example 7.1
Listing 7.1: Using a Python interpreter to illustrate how some datatypes in Python can be casted to others, but not not the other way around.

```python
$ python3
Python 3.5.2 (v3.5.2:4def2a2901a5, Jun 26 2016, 10:47:25)
[GCC 4.2.1 (Apple Inc. build 5666) (dot 3)] on darwin
Type "help", "copyright", "credits" or "license" for more information.
>>> my_int = 3
>>> int(float(my_int))
3
>>> int(float(my_int)) == my_int
True

>>> my_float = 3.33
>>> float(int(my_float))
3.0
>>> float(int(my_float)) == my_float
False

>>> float(str(my_float))
3.33
>>> float(str(my_float)) == my_float
True

>>> my_str = '3.3333333333333333333333333333333333333333333'
>>> str(float(my_str))
'3.3333333333333335'
>>> str(float(my_str)) == my_str
False

>>> str(int(my_str))
Traceback (most recent call last):
  File "<stdin>", line 1, in <module>
ValueError: invalid literal for int() with base 10: '3.3333333333333333333333333333333333333333333'
```

In the Python language, the value None is a lack of value. Therefore, the different types of values are given the following intensities:

- None: 0 %
- Integer: 55 %
- Float: 65 %
- String: 94 %

All values are a percentage of the maximal intensity 1. After the first allocation of a value, the element’s value is smoothed out by computing the average value across the element’s eight adjacent neighbours.
The following techniques are the options in the decision of clustering method. They are always employed separately. The clustering phase results in a list of hypotheses objects.

- Felzenzwalb image segmentation algorithm
- Mean shift clustering
- DBScan clustering
- Agglomerative clustering

When using the Felzenzwalb image segmentation algorithm, the image is first created using the datatype to pixel value mapping. Segments of the image are given as output, and by this the global blocks are to be identified. The segments in the image is called clusters in the other algorithms.

7.3.2 Cut the dendrogram

When using the agglomerative clustering method, the last step of the method is to choose at what level to cut the dendrogram in order to obtain the final clusters. How to decide where to cut the dendrogram is a hard problem to solve. Let us be reminded of the fact that we do not need to find the optimal cut directly, but can instead do several cuts and let each cut represent a new hypothesis. Therefore, as of now, three different cuts are made. This number can of course be increased if none of these are satisfactory.

The second order difference $D''(A, B)$ of the linkage $D(A, B)$ given by

$$D''(A, B) = D''(A, B; x) = \frac{D(A, b; x + h) - D(A, b; x)}{h} - \frac{D(A, b; x) - D(A, b; x - h)}{h} = \frac{D(A, b; x + h) - 2D(A, b; x) + D(A, b; x - h)}{h^2}$$

where the $D(A, B) = D(A, B; x)$ and thus $D''(A, B) = D''(A, B; x)$ are dependent on the cut $x$. We have chosen to cut where $D''(A, B)$ of the linkage $D(A, B)$ is the highest. This is shown to give good results for at least one of the choices of linkage, the single linkage. In Figure 7.3 $D(A, B)$ and $D''(A, B)$ are plotted for the last twenty merges. The two largest $D''(A, B)$ are marked as knee points. The resulting clustering using these two cuts can be seen in Figure 8.6 in Section 8.

---


7.3.3 Extracting relations

After finding blocks, the next step is to extract relations between them and labels around them.

Firstly, for each block, create a dictionary of label-value pairs. For each string element, assign a non-empty element either directly beneath it or directly to the right as its value in the pair.

Secondly, for each block $b$, look at the blocks directly above it. If any of the above blocks contains any label-value pairs (i.e. it contains at least one string element) these label-value pairs are added as labels associated with $b$, for example \{‘Plate’: 1, ‘Barcode’: ‘123456’\}.

The current algorithm only considers blocks consisting of numbers when finding these labels. This is a clear limitation of the algorithm, as it only considers a subset of all blocks. Another limitation implemented in the current algorithm is that it searches for labels only in the positions above the block, and not to the left, right or below. Above the block is chosen because it fits the victor 3 file format.

The next step is to find the set of unique labels. This set is considered global because it is associated with the hypothesis object, in contrary to the label-value pairs which are associated with each number-block. For each unique label, all the values are listed together. For example, if the label ‘Plate’ is found in five different locations and have five different values, and another label ‘Repeat’ is found at five different places but only have four different values, the result would be \{‘Plate’: [1, 3, 4], ‘Repeat’: [1, 2, 3, 4]\}.

In the end of the hypothesis generation step, each hypothesis consists of one labeltree and one nnd array. These objects are created in this step. When a labeltree object is created, a dictionary called label_to_dimension_dict is created which maps labels

---

(string) to dimensions (integer). Examples of a label are "row", "column", "Plate". In the labeltree, these labels are seen as dimensions.

A node object is created for each unique label. Along with the unique label, the node object also holds its corresponding unique dimension integer.

Then, for each value associated with the unique label, a child node is created that holds the value along with a unique integer. To exemplify, the label row could have values 0, 1, 2, 3, ... and the label Start temperature could have values 15.3, 21.1, 19.9. The first value of the label Start temperature is 15.3, for which a child node object is created and given a unique integer called index dimension. Technically, each node can store a value, a pointer to its parent, and a list of pointers to its children. The pointers make it possible to traverse upwards and downwards in the tree. See Figure 7.4 for the example mentioned above.

![Figure 7.4](image)

**Figure 7.4:** An example of a simple labelTree showing the first four nodes. The top two have two children each.

The next step is to create the corresponding ndn array for each hypothesis. Blocks containing string values are neglected in order to simplify the algorithm and reach a result faster. This because if the algorithm is allowed to only process blocks containing numbers, we can assume that each block holds measurement data for a microtiter plate of size 96, 384 or 1536. Thus, in the current algorithm, logic is hardcoded to assign the well id and the relative position of each of the element in the block. The well id is a string and is derived from the relative row and relative column of the element. For example, in a microtiter plate with size 96 (8 rows and 12 columns), the well id:s first row ranges through A01, A02, ..., A12, thus building the well id:s shown in Equation 7.1.

\[
\begin{align*}
A01 & \quad A02 \quad \ldots \quad A12 \\
B01 \\
\vdots & \quad \ddots & \quad \vdots \\
H01 & \quad \ldots \quad H12
\end{align*}
\]  

(7.1)
7.4 Hypothesis evaluation

How to evaluate a hypothesis? How do one recognise a good hypothesis? It is not clear, and the result depends heavily on what is included in the objective function. The following metrics are used to evaluate each hypothesis, together with some comments on how they could be used or why they are significant:

- **instrument**: The type of instrument used for the measurement of the experiment, which has generated the input file.\(^7\)

- **filename**: The filename contains information about what type of experiment etc.. This is not used in this study.\(^7\)

- **clustering method**: Clustering method used in the hypothesis generation step.

- **n_blocks**: Total number of blocks found in the hypothesis step, both blocks only containing number elements, blocks only containing string elements, and blocks only containing both number elements and string elements. If the number of blocks is known this metric is an easy way to see which hypothesis seems to be correct.

- **n_row**: Total number of rows found in the hypothesis step. In other words, the maximum row index found plus one. This metric could be used to early screen out poorly performing hypotheses, for example those which discard too many elements and therefore loose information during the process.

- **n_col**: Total number of columns found in the number blocks. This metric could be used to early screen out poorly performing hypotheses which for example discard too many elements.

- **n_str**: Total number of string elements found. This metric could be used when choosing clustering method.

- **n_int**: Number of integer elements found. Same usage as above.

- **n_float**: Number of float elements found. Same usage as above.

- **col_of_most_freq_min_range**: The index of the column in the number blocks in which the smallest range among all values in that column was found the most times.

- **n_col_most_freq_min_range**: Number of columns having the same index as the index col_of_most_freq_min_range.

- **row_of_most_freq_min_range**: The same metric as above but looking at rows instead of columns.

- **n_row_most_freq_min_range**: The same metric as above but looking at rows instead of columns.

\(^7\)In this first version of the algorithm, this piece of data is given by the user.
• **col_of_most_freq_max_range**: The same metric as above but using largest instead of smallest range.

• **n_col_most_freq_max_range**: The same metric as above but looking at rows using largest instead of smallest range.

• **row_of_most_freq_max_range**: The same metric as above but looking at rows and using largest instead of smallest range.

• **n_row_most_freq_max_range**: The same metric as above but looking at rows and using largest instead of smallest range.

• **most_freq_rowdiff**: The most frequent difference in rows between two blocks/clusters.

• **sec_most_freq_rowdif**: The second most frequent difference in rows between two blocks/clusters.

• **third_most_freq_rowdif**: The third most frequent difference in rows between two blocks/clusters.

• **n_96_blocks**: Number of blocks containing 96 elements.

• **n_384_block**: Number of blocks containing 384 elements.

• **n_1536_block**: Number of blocks containing 1536 elements.

• **n_96_number_blocks**: Number of blocks containing 96 elements of type integer or float.

• **n_384_number_blocks**: Number of blocks containing 384 elements of type integer or float.

• **n_1536_number_block**: Number of blocks containing 1536 elements of type integer or float.

• **n_lost_element**: Number of elements discarded during hypothesis generation until after creating the nnd.

• **n_elements_lost_before_nnd**: Number of elements discarded during hypothesis generation until before creating the nnd.

• **n_blocksize**: Number of distinct block sizes.

• **most_frequent_blocksize**: The most frequent block size.

• **second_most_frequent_blocksize**: The second most frequent block size.

• **third_most_frequent_blocksiz**: The third most frequent block size.

• **n_parents_in_labeltree**: Number of parents to leaf nodes in the labeltree.
The reason behind using ranges when creating metrics is because we want to find the control rows and columns in the data from a microtiter plate. Remember, there are almost always one group of negative control wells and one group of positive control wells in a single plate. The controls are usually placed as columns, usually as the far right columns. But the negative and the positive control group could also be placed in the same column, one group above the other one.

The purpose of a negative control well is to display the substrate when it does not respond to the experiment treatment. The purpose of a positive control well is to display a substrate which responds fully to the experiment treatment. Together, they are used as references to the other wells where the experiment treatment of interest actually happens. For example, the negative control wells display a response of near 0 percent and the positive control wells display a response of near 100 percent. One can then expect the other wells to display values ranging between 0 and 100 percent. These two values can also be used to normalise the other values. Outliers outside of this range could be thrown away.

7.5 Learning

The learning step was planned to come after the hypothesis evaluation step, but due to lack of time, it is not implemented. The following describes this theoretical step.

The goal of the learning step is to update the weights $w$, by solving the optimization problem

$$\max_x f(x,w) = A(x)w$$

subject to $g_i(x,w) \leq 0, \ i = 1, \ldots, m.$

$h_i(x,w) = 0, \ j = 1, \ldots, p.$

Let $x$ represent the input signal to the algorithm, i.e. the input file. Let $A$ be the matrix given by the output of an input file $x$ in the hypothesis evaluation step described in Section 7.4. Each column in $A$ is a metric, and each row is a hypothesis. The weights $w$ are used to determine how much to give weight to the different metrics. The functions $g_i(x), i = 1, \ldots, m,$ and $h_i(x,w), j = 1, \ldots, p,$ set constraints to the optimization problem.
Chapter 8

Result – The Hypothesis-generating Approach

In this section, we present the results from the approach presented in Chapter 7. It is an algorithm that generates different hypotheses on how to interpret the spreadsheet.

![Table](image)

Figure 8.1: Example of output generated by a hypothesis. It is presented in the preferred format in Figure 1.3 and described in Section 4.1.4

Figure 8.1 is the result of one of the hypotheses of the algorithm. The format in this hypothesis is identical with the one the user want to have. This is a case where the algorithm gives excellent results. Let us examine when it does and when it does...
not perform well.

Figure 8.2: Example of input that the algorithm can correctly interpret. It is from the sheet called Plate in the file Abs 630 nm - 384 well - 0.1s12000002201512161335.xls, which is of file type Victor 3.

The result is produced when running the algorithm on the file Abs 405 nm0200001710471820140416.xls, which is of type Victor 3. The algorithm only considers the sheet called Plate. The result comes from the hypothesis with clustering method agglomerative with single linkage and first cut. As a reminder, Linkage is a clustering parameter deciding what measurement of similarity the clustering method should have.

In fact, the algorithm has been fine-tuned to interpret sheets called Plate in files of the type Victor 3. It therefore gives excellent results for any sheet in a file of type Victor 3, even those which have incomplete data. Another example is the file Abs 630 nm - 384 well - 0.1s12000002201512161335.xls. Figure 8.2 presents a screenshot of the sheet Plate of this file. In the sheet, there are only 3 × 6 values. Compare that with the usual sizes of 8 × 12, 16 × 24, or 32 × 48 normally found in a Victor 3 file. Note that all the meta-data names and values are still in the input file, please see row 1, 2, 4 and 5 in the input file. In Figure 8.3, one of the interpretations of the data in the input file is presented. The interpretation is made by the same hypothesis as above, the one with agglomerative clustering, single linkage, and first cut.

Please note the column with header well_id. This column is specific to Victor 2 and Victor 3 files, and is derived from which row and column the value is found, relative to the first value in the block. For instance, the value in the first row (index 0) and first column will have the well ID A01 where the letter represents the row and the number represents the column. Similarly, the value on the third row (index 2)
and sixth column (index 5) is given the well ID C06. There is no information about well ID in the input sheet, it is purely generated based on domain knowledge.

Figure 8.3: The output of the sheet presented in Figure 8.2. All values on rows 7, 8 and 9 in the input file are transformed into the column L in the output file, and each of these values are annotated with meta-data coming from the meta-data names and values in row 1 respective row 2 in the input file. The interpretation is done by the hypothesis using agglomerative clustering, single linkage and first cut.

The algorithm is run on dataset 1, which contains in total 51 files of filetypes Envision, Microbeta, Victor 2 and Victor 3, see Tables C.1 and C.2 in Appendix C for the complete list. For each file, 15 different hypotheses are generated, see Figure 7.2, and each hypothesis is evaluated using the metrics listed in Section 7.4.

Three of the most interesting metrics are the ones counting the number of blocks found of size 96, 384, or 1536 and only containing numbers. If such a block is found, it most probably is a microtiter plate. The Figure 8.4 presents the number of blocks of size 96 only containing numbers \( n_{96\text{\ number\ blocks}} \) per filetype and clustering method. Applied on Victor 3 files shown in yellow, we can see that the agglomerative clustering method with single linkage clearly performs the best. When applied on Victor 2 files shown in grey, the agglomerative clustering method with average linkage performs the best. It is notable that the algorithm does not find any block of size 96 containing only numbers in either Micrometa files nor in Envision files.

When looking at the equivalent metric for 384, there are only blocks in the Victor 3 files. The only method that can find these blocks are the agglomerative clustering method with single linkage, which finds 79 in total. There is no file in the dataset which contains microtiter plates of size 1536. As a consequence, none of the hypotheses have a non-zero score on this metric.

8.1 Agglomerative clustering

One step in the algorithm is to cluster the cells in the sheet into different blocks. What we want to achieve in this step is to separate cells containing data from the rest, such as cells containing labels (titles, names of dimensions, etc).
As one can see in Figure 8.5, agglomerative clustering with single linkage and first cut is the hypothesis having the best result in terms of the metric most_frequent_blocksize. How the dedrogram is cut is explained in Section 7.3.2. Every agglomerative clustering produces a dedrogram, an example is shown in Figure 8.5 done using the hypothesis parameters mentioned above on a sheet called Plate in the file ‘AMC fluorescence02000023124720150702.xls’. In this example, the cut in the dedrogram produces 9 clusters different colours. Figure 8.6a is produced with the same hypothesis. It shows cells in the sheet, coloured according to which cluster they belong to. For comparison, Figure 8.6b shows the respective second cut, which gives 6 clusters.

Agglomerative clustering seems to be an effective method to find blocks of cells separated by a sea of empty cells. At the level in the dendrogram where separate blocks are agglomerated together, there will also be a spike in $D''(A, B)$ as seen in Figure 7.3.
Figure 8.6: Agglomerative clustering with single linkage, first and second cut, resulting in 9 and 6 clusters respectively. This is the result from clustering on the sheet called Plate in file ‘AMC fluorescence02000023124720150702.xls’.
Chapter 9

Discussion

In Section 9.1 the result of the unsupervised learning is discussed, and in Section 9.2 the result of the hypothesis-driven algorithm is discussed.

9.1 The Unsupervised Learning Approach

In Chapter 6 we present three experiments on three datasets. None of these give satisfying clustering results.

We only tried one clustering method, which is the mini-batch K means clustering method. This clustering method use Euclidean distance. In Section 4.4.1 we describe problematic phenomena that can occur in a high-dimensional space when using Euclidean distance. Therefore, we suggest to try different clustering methods in future studies. Ideally methods for which one can specify the distance metric. Then, one can try different distance metrics to figure out which one is the most suitable for given dataset. Klawonn et al. (2015) shows that it is difficult to know in advance which value on $p$ in the norm $L^p$ that will give the best performance on a particular dataset.

In future studies, one could also try supervised learning. The labour needed to mark up each sheet called Plate is possible to do. It would also be fairly straight forward because the work of deciding how to label each cell has already been done in this study. The labels can be given in spreadsheets. Write the labels name in a separate sheet where the label for each cell has the same position as its corresponding cell. Then read the sheet in the same way as the original sheet. One idea that came up early in the discussions with Thomas about the desired solution is to let the user label each cell. When that is facilitated, one can use a learning process where new labels are iteratively added to the dataset by the user. First, the unlabeled data samples are clustered. Then presented to the user, and the user can edit and correct the result. Then this could be done in several iterations, where unlabeled samples are labeled by the user.
One important output of this study is the pipeline built and described in Chapter 5. The pipeline can read a number of .xls, .xlsx, and .csv files and transform them into a standard feature matrix. When on this form, one can easily apply any machine learning model because the data is in a standard format for machine learning analysis, i.e. each row is a sample and each column is a feature, where a sample in this case is considered to be a cell.

Another way of interpreting the data is to consider each sheet, or even each file, as its own sample. This interpretation is closer to the one in, for example, document clustering and document classification. This might also be closer to what we want to achieve with this study. The structure we want to find is something inherently specific to each sheet or file. That fact is a good incentive for considering each sheet or file as its own sample.

A first approach could be to not use as many dimensions as we have used in this study. One could settle with one-hot encoding dimensions for each data type: integer, float, string. Then, the complexity of each sample would be in the same order as a three-channel image, which is a standard case in both supervised and unsupervised learning.

### 9.1.1 Dimensionality Reduction

As noted in Section 6.1.1, all of the four dimensionality reduction methods perform almost the same throughout components 1 to 200. This holds for all tree experiments, that is on all three datasets. The results suggest that there are no difference in performance between the four methods.

When we only read sheets called Plate into the dataset, fewer components are needed to keep most of the variance after the dimensionality reduction compared to if we read all sheets in the input files. This might not come as a surprise. Because the format in these sheets are the same, the variance should be smaller.

One future approach would therefore be to first group together the data into different distinct types of formats. For example, group together each cell found in sheets or files similar to the sheets called Plate. Both the name of the sheet and the name of the file are already stored as one-hot encoded dimensions in the feature matrix. Therefore, one could use those features as labels and use supervised learning to classify different formats. Natural Language Processing might be required to preprocess the name of the sheets and files.

A way to reduce the dimensionality would be to introduce new datatypes, for example a datetime datatype. In this case, all cell values identified as something symbolising a date, a time, or a combination, could be labeled as the datatype datetime instead of string. Then, these cells could be given the value 1 in a column called datatype.datetime. This would hinder each unique one of them to have its own string-column starting with s... That means we would have one more dimension instead of a number of dimensions equal to strings representing unique datetime. One could even be more specific and distinguish between cell values representing
Unsupervised learning was chosen because we did not want the hassle of labelling the data. But instead, we were punished to only use some tools for evaluating our clustering results, and these turned out to perform poorly, see discussion about the Davies-Bouldin index in Section 6.2.

9.1.2 Evaluating the Clustering

In Scikit-Learn’s implementation, both the Calinski-Harabaz index and the Davies-Bouldin index are computed by using the Euclidean distance. This is problematic because of the high dimensionality of our data, see discussion in Section 4.4.1.

Our implementation of Matthew’s correlation coefficient was not enough computationally efficient. In future studies, one could instead implement a multi-class version of Matthew’s correlation coefficient based on Equation 2 in the paper of Jurman, Riccadonna and Furlanello (2012). It may be practically more scalable. We can also note that the poor performance is coupled with the size and dimensionality of the dataset. One might be able to use the present implementation of Matthew’s correlation coefficient on lower-dimensional data.

9.1.3 Scaling

The choice to implement the preprocessing and machine learning using NumPy, Pandas and Scikit-Learn as single-thread application is probably not scalable.

We underestimated the number of cells there are in spreadsheets, and the number of dimensions given by using one-hot encoding. Because the dataset is large, it is hard work when using only one core or Python thread. A better choice might have been to implement everything in a cloud platform, such as Amazon Web services, Google Cloud Platform, or Microsoft Azure, and leverage on parallelized and distributed computing. Then, one could read a larger dataset, which makes the results more generalizable. Then, it should also be possible to use machine learning frameworks which are native to distributed-computing.

9.2 The Hypothesis-generating Approach

The following discussion is specific to the result of the hypothesis-generating approach.

Our intention is to build an algorithm which is as generic as possible. We wanted to make as few decisions as possible that would narrow down the different types of formats of input it can handle. For example, we wanted to build something that would use machine learning, and did not consider a rule-based algorithm to
be generic enough. The reason to not have hard-coded rules and logic inside the algorithm is because the algorithm’s ability to learn would be confined by these hard coded rules. But due to limitations in time, we needed to limit the scope, which meant to build in rules and logic into the algorithm. An example of a built-in rule is how it finds a value to a key when building up a key-value pair; it only looks at the cell below, and the cell to the right.

The algorithm was also fine-tuned against the Victor 2 and Victor 3 file types. Both the rules and the metrics are crafted in order to facilitate these two data formats, specifically the sheet called Plate. Therefore, the algorithm as it is now should be considered a build-to-learn project. And we indeed learnt from building this algorithm.

9.2.1 Learnings

The general idea of the architecture might still be plausible to generate hypotheses which one then evaluate and learn from. The missing step is learn, which we did not have time to implement. As discussed in Section 9.1, one idea for the unsupervised learning approach would be to let the user label the data. This could be a feasible idea for the hypotheses-generating approach as well. In this case, the user could score and rank the hypotheses, and maybe even modify the best one to make it even better.

Silva et al. (2006) proposed using back loops between the different stages of the interpretation process. They suggest high interactions between the different steps in order to reduce errors, revisiting precious decisions under the light of new information to increase the confidence in the decisions already made. This is not currently implemented, but should definitely be considered in future work.

We failed to have a clear cut between the algorithm and the format of the output data, which is the labeltrees and ndd arrays. This coupling sets unnecessary constraints on the algorithm. The idea behind using custom-made data structures could also be discussed.

One way to improve the metrics in section 7.4 could be to re-design them to be linear with the “goodness” of the score: the higher the score the better. An attempt is already made to do so as some of the metrics are just derivatives of another metric, but counting it in some way. For instance, n_col_most_freq_min_range is just a derivative of col_of_most_freq_min_range. It counts the number blocks in which the column-wise most frequent minimum range is found in the same column index. The derivative metric makes it easier to evaluate the hypotheses – higher score means more conformity between the data in the number blocks. It is also an indication that we have found microtiter plates, because the positive and the negative control columns are usually positioned in the same columns in every microtiter plate.
9.3 Future work

When only considering sheets with the name Plate from a Victor 3 file, each cell data type should be dependent on its position. This because the data follows a clear structure, see for example Figure 4.1. Independent on the number of plates, one can always assume that there is a row of label names first, then label values, then an empty row, then a single label name and its value, then an empty row, and the plate of size 96, 384, or 1536 cells. Mathematically, this means that the data type dimensions should be linearly dependent on the position dimensions. For future work we therefore suggest to train a model that predicts if the cell is a float (value 1 in column datatype.float), an integer (value 1 in column datatype.int), or a specific string (value 1 in a column starting with s). The prediction should be based on the position features (row and column) and which file and sheet the cell lives in.

We also suggest to investigate the value of adding document classification as a pre-step before the actual step of understanding the structure in the spreadsheet. One way would be to cluster sheets or files based on the frequency of the words in them. This is a standard problem, solved by many others. Another way would be to cluster on the format of the data, instead of the actual domain of the data. When processing Victor 3 files for example, then one can group all sheets called Plate together, all sheets called Protocol together, all sheets called List ; Plates 1 - XX together etc. Then, these different types of spreadsheets can be regarded separately. One would also be able to understand what approach or technique would work on what data type. Another approach would be to parse all spreadsheets as if they would be text documents.

A completely different approach would be to try to use an autoencoder to find inherent features in the format of spreadsheets. An autoencoder is a type of artificial neural network used in an unsupervised fashion to learn to represent high-dimensional data in a lower dimensional space. This can be used for dimensionality reduction, but also to learn inherent features of the dataset.
Chapter 10

Conclusion

With this thesis, we try to address the problem of interpreting intrinsic schemas in spreadsheets. Three contributions are made towards the long-term solution described in Section 3.1.

The first contribution is aimed to be general in its applicability. This, by evaluating the use of unsupervised learning to cluster cells in spreadsheets. The clustering is mainly based on the cells’ positions, datatypes and values. No satisfying results are found by using the mini-batch K-means clustering algorithm. When only looking at a specific data format – the format in sheets called Plate in the Victor 3 files – 95 percent of the total variance can be explained by only using around 40 dimensions. This is significantly lower number of dimensions compared to the original number 542. Future research is welcomed to leverage on this study’s preprocessing tools.

The second contribution is aimed to be short-term. The by building a hypothesis-generating algorithm to provide different hypotheses on what schema there is in a given sheet. This algorithm is adapted to the specific filetypes Victor 2 and Victor 2, which translates to a specific schema. The algorithm works well on this particular schema, but is not generalisable because of its built-in rules. These rules are specific to the schema. Many learnings are made during the process. Future research is welcomed to leverage on these learnings.

The third and final contribution is an initiation of a discussion. We start to outline a holistic SaaS solution, by providing discussion around key choices of design and technologies. Due to the rapid evolution of software technologies, some parts of these discussions are no longer applicable, as newer technologies or design choices that did not exist when the study was undertaken might be more relevant today. Some discussions made in this study can arguably still be valid.
Bibliography


Chen, Z. and Cafarella, M. (2013). Automatic web spreadsheet data extraction, 
Proceedings of the 3rd International Workshop on Semantic Search Over the Web, 
SS@ ’13, ACM, New York, NY, USA, pp. 1:1–1:8.

overview, Fundamenta Informaticae 2005: 1001–1038.


Codd, E. F. (2009). Derivability, redundancy and consistency of relations stored in 


Dasu, T. and Johnson, T. (2003). Exploratory Data Mining and Data Cleaning, 1 
edn, John Wiley & Sons, Inc., New York, NY, USA.

Django Documentation

Django Documentation (2016).

identifying and interpreting tables in plain text, Proceedings of the Fourth Annual 

Durrant, R. J. and Kaban, A. (2009). When is ‘nearest neighbour’ meaningful: A 

paradigms: a research survey, International Journal of Document Analysis and 
Recogntion (IJDAR) 8(2): 66–86.

Embley, D. W., Tao, C. and Liddle, S. W. (2005). Automating the extraction 
of data from html tables with unknown structure, Data & Knowledge Engineering 
International Conference on Conceptual Modeling.

for discovering clusters a density-based algorithm for discovering clusters in large 
spatial databases with noise, Proceedings of the Second International Conference 
on Knowledge Discovery and Data Mining, KDD’96, AAAI Press, pp. 226–231.

extration from documents, IBM Journal of Research and Development. cited By 
0.


Flask Documentation

Gunicorn Documentation


Microsoft Remote Procedure Call

Microsoft Remote Procedure Call (2016).


Retrieved 2016-05-05.


Nginx Documentation

Nginx Documentation (2016).


Retrieved 2016-06-09.

paradigm4

paradigm4 (2016).


Retrieved 2016-05-09.


The word "structure" in the Cambridge Dictionary Online


uWSGI Documentation website

uWSGI Documentation website (2016).


W3Techs.com

W3Techs.com (2016).


What is Docker?

What is Docker? (2016).

Who is using scikit-learn?


Appendix A

Key choices in design and technologies

In the following section, a discussion will be brought out on how to develop the SaaS solution in which the dataloading component will be encapsulated. Findings on relevant best practice of software development is presented, together with motivations on the chosen solution components. Please note, the research undertaken for this chapter was done during the year 2016, and presents what was relevant during that time. These some of these areas are evolving rapidly, and new technologies are replacing old ones.

A.1 Overview of the system

The following pipeline for the solution is proposed:

1. File upload by the user to the website.
2. File format analysis.
   (a) Locate target data and surrounding metadata (i.e. labels such as plate number, measurement time, barcode).
   (b) Feature generation/extraction — generate a large numbers of features from data
   (c) Feature selection — reduce the number of features
   (d) Generation of structure hypotheses
4. Scoring of hypotheses. The data structures and meta-data are compared with previously built machine learning models. Statistical modelling are used to score and prioritise the hypotheses.
5. Visualisation of the highest scored hypothesis. Here, the user will be able to edit the result or reject the hypothesis in favour for another one. Various probability density functions and histograms might also be presented when applicable.

6. User chooses one hypothesis and gives feedback.
   (a) Storage of meta-data and feature vectors of the hypothesis (multi-dimensional arrays) selected by user including its edits. (Actually removal of the rejected ones.)
   (b) Translation of user edits to structural changes and updating of scoring algorithm. This is the training of the algorithm a la supervised learning.

7. User export of data matrix.

A non-exhaustive set of choices for how to design the system are described in this chapter, many more are left out because of time limitation. The following are the choices recommended when building the final solution:

- System design patterns: REST
- Web development framework: Django
- Database: PostgreSQL
- Web server: Nginx
- Load balancer: Nginx
- Web Server Gateway Interface: Gunicorn
- Containerization: Docker

A.2 System Design Patterns

A.2.1 REST

Representational State Transfer (REST) is chosen as the architectural style of the system and the Application Programming Interface (API) of the components in the system. This because it is easy to implement and the style it widely spread among developers. Then, future developers of the system will easily be able to understand the style and system. Also, Django projects often follow a RESTful design, which contributes to the decision.

REST is the architectural style of the world wide web, developed by Fielding and Taylor (2000). It is a set of coordinated constraints on how servers, clients, data
elements and other components communicate with each other, and how the client
browses through a hypermedia system.

REST could be seen as the API of the system in the sense where REST ignores the
implementation of the components and instead focuses on the roles, the constraints
of their interactions, and their interpretation of data elements. The system built
in this project can be called RESTful as it follows the constraints of the REST
architectural style (Fielding and Taylor, 2000):

1. **Client-server division**: A clear separation and a uniform interface between
   the client and the server allows servers to evolve independently of the client,
   separates the client from concerns of data storage on the server-side and
   thereby allowing portability of user interfaces across platforms. Furthermore,
   it improves scalability by simplifying the server component.

2. **Statelessness**: Stateless communication in the sense where all the informa-
   tion needed to interpret the message is included in the communication, and
   therefore the state of the server is not used to interpret the message. When the
   server does not need to keep track on which state it is in, the system will see
   simplification of the communication and therefore scalability, and improved
   visibility. A drawback is an increased per-interaction overhead because all the
   information needs to be sent in each request.

3. **Cacheable**: A response from a server to a client could be labelled as either
   cacheable or non-cacheable which allows the client to reuse data for future
   equivalent requests.

4. **Layered system**: This constraint allows for intermediate components be-
   tween client and server, which could allow for load balancing, providing shared
   caches, or enforce security policies.

5. **Uniform interface**: This constraint is fundamental for any RESTful system.
   It simplifies the architecture and enables the components to evolve indepen-
   dently. The four sub-constraints are

   (a) **Identification of resources**: Each piece of data, i.e. resource, is uniquely
       identified using its Uniform Resource Identifier (URI).

   (b) **Manipulation of resources through these representations**: The
       resource itself is not transferred to the client, but instead a representation
       of it. This representation contains all the information needed for the client
       to manipulate, modify and delete the actual resource on the server side.

   (c) **Self-descriptive messages**: Each message contains all the information
       needed for the receiver to parse and interpret it.

   (d) **Hypermedia as the engine of application state (HATEOAS)**:

---

1Hypermedia is described as multimedia with hyperlinks to other elements of media, forming
a system on non-linear multimedia. It was first mentioned in Nelson (1965), and the world wide
web is the obvious example of a hypermedia system.
6. **Code on demand (optional):** By transferring executable code to the client, the server can extend or customize the functionality of the client. Most notably is the client-side scripts, such as JavaScript, but also compiled components such as Java applets could be transferred.

Because one only needs to follow the mentioned set of constraints, it is easy to follow it using the Hypertext Transfer Protocol (HTTP). Then, the HTTP request methods (GET, HEAD, POST, PUT, DELETE, TRACE, OPTIONS, CONNECT and PATCH) are already used by web browsers to request html pages and send data to web servers.

REST is usually compared with Simple Object Access Protocol (SOAP) when choosing the way to communicate between client and server. This, even though REST is an architectural style and SOAP is a protocol. As web browsers already use REST, and to follow the style one needs to follow a set of constraints. To compare, using SOAP seams to be more complex.

### A.2.2 Modularity

If a system is modular, with clear cuts between different components, several things are achieved. It is easier to develop and maintain a part of a system if it is clearly separated from the rest of the system. It is also easier to test, because if there is a clear interface, it is easy to test in an input-response fashion. Clear separation also gives a better understanding of what dependencies each component has. It also enables for re-use of a component inside the system.

### A.2.3 Application Programming Interface

In general terms, an application programming interface (API) is an interface of a software application, and specifies how other applications can communicate with it. Sometimes an API is described as a library, i.e. a set of callable functions and methods used by other programs. Then, the API describes expected functionality and behaviour without revealing its implementation. Sometimes, the interface between components inside the same system is also called API.

Mentionable examples are the APIs of the operating systems, which defines how applications run on their operating system and use its user interface (see for example [Windows API Index (2016)](https://msdn.microsoft.com/en-us/library/windows/hardware/jj658319(v=vs.85).aspx), or the API of Sci-kit Learn ([Buitinck et al.; 2013](http://scikit-learn.org/stable/)). For the latter case, the API is to be found here: [http://scikit-learn.org/stable/modules/classes.html](http://scikit-learn.org/stable/modules/classes.html).

In distributed computing, a remote procedure call (RPC) is a call on a function or method in a program which causes a function or method to execute on another computer on a shared network, which is coded as if it were a local function call ([Birrell and Nelson; 1984](http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.35.6859&rep=rep1&type=pdf)). In a client-server communication, the client makes a
request call to the server, a RPC, and the server executes the call and replies the client with a response call (Microsoft Remote Procedure Call [2016]).

A.2.4 Server-side Web API

A server-side web API is an interface of a web server, exposed to the World Wide Web by a defined request-response messaging system. This is usually done by using the transport protocol Hypertext transfer protocol (HTTP) and JSON or XML as means of the communication.

A.3 Web Development Framework

There are a myriad of web frameworks available on the market. Features including the licence (free or proprietary), the programming language it uses, the ease of installation, database abstraction and Object Relational Mapper (ORM), and the learning curve of the new developer. For this project, the Python-based Django framework is chosen as the web development framework as it is free and open-source, and one of the most used web frameworks in the world. It therefore has a large community and extensive documentation. It is distributed with several useful tools and modules included which allows for a fast development and early deployment. To mention three of them: django.contrib.admin, the automatic admin site which provides a quick, model-centric interface for trusted users to manage content; django.contrib.auth, the user autentication system which handles user accounts, groups, permissions and cookie-based user sessions; django.contrib.postgres, a module of features specially for the database management system (DBMS) PostgreSQL (Django Documentation [2016]).

Flask is another web framework often mentioned at the same time as Django. Flask is also written in Python, and is considered to be a light-weight framework and does not provide as many built-in modules and tools compared to Django (Flask Documentation [2016]). Flask might be better suited for highly customized use cases, or in cases where performance is a key factor.

In Django, models are used to manipulate data in the database. A model is a Python class that inherits from django.db.models.Model and maps to a table in the database. The attributes of the class maps to columns in the table and an instance of the class is represents a tuple in the table. Django facilitates for automatically generated access to the database when using models, therefore there is no need of coding SQL queries.
A.4 Web Server

As of June 2016, the Apache HTTP Web server and Nginx are two of the most popular Web servers on the market (W3Techs.com 2016) for handling HTTP requests from web browsers, please see Figure A.1. Nginx is chosen for this project because of its small usage of server resource and its popularity among sites which need to handle large number of requests simultaneously (Nginx Documentation 2016).

Figure A.1: Percentages of websites using various web servers, as of June 2016 (W3Techs.com 2016). Please note that a website may use more than one web server.

A.5 Interface between the Web server and the web application

Django uses WSGI as its primary deployment platform (Django Documentation 2016). WSGI stands for Web Server Gateway Interface and is the Python standard for Web servers and applications, i.e. “a specification that describes how a web server communicates with web applications, and how web applications can be chained together to process one request” (WSGI Documentation 2016).

Gunicorn is chosen as the WSGI HTTP server for the project. It has no dependencies and natively supports Django (Gunicorn Documentation 2016).

Another comparable WSGI is uWSGI, which is included in the official Nginx distribution (uWSGI Documentation website 2016). On its documentation website, uWSGI is promoted as having the strengths of versatility, performance, low-resource usage and reliability (uWSGI Documentation website 2016).
A.6 Database

A.6.1 SciDB

SciDB is a database management system (DBMS) where the data is stored as multidimensional arrays within each cell. It is design to use commodity hardware as nodes in a distributed system, allowing for scalability. It is also designed to store both dense and spares data, and its novel data model suggests a better fit with data from sensor, sequencing and time-series data compared to the more traditional relational data model. This by having data that is close to each other in the user-defined coordinate system also stored together in the database. SciDB supports in-database analytics, allowing to do cost-effective operations on the data on the DBMS-side instead of doing the analytics on incomplete or costly queried data on the cient side. SciDB Community Edition is freely distributed under an open-source licence[paradigm4; 2016].

A.6.2 PostgreSQL

Complies with SQL standards, unlike MySQL which has diverted away from the standards. PostgreSQL is therefore better suited if the data at some point will be emigrated to Oracle-like databases. PostgreSQL is ACID compliant, which makes it more reliable compared to a non-ACID compliant DBMS, for example MySQL. PostgreSQL is not an option to work in conjunction with MySQL.

A.7 Containerization

Docker is used for containerization of the software. Docker is an open-source software that makes it easier to deploy software. By putting the software in containers, one can guarantee that the program always runs the same, regardless of the environment the container runs in. This is done by wrapping the piece of software into the container together with everything it needs to run, including code, runtime, operating systems, system tools, system libraries and other installable programs. “Containers have similar resource isolation and allocation benefits as virtual machines but a different architectural approach allows them to be much more portable and efficient.” [What is Docker? 2016]

A.8 State transition diagram

In Figure A.2, a state transition diagram of the system is presented. It shows how the user can travel from different states.
Figure A.2: State transition diagram of the system from a user perspective.
Appendix B

How is the problem handled today?

The problem of handling tabular data in various formats is not new to the researchers at CBCS. Today, they have a few methods for handling the problem (Thomas Lundbäck, personal communication). The most simple method is by using Microsoft Excel to copy and paste cells into the preferred format. This is time consuming and is only time efficient for small datasets. For datasets with more than a handful files, it takes too long to be an alternative. It is also error pruning because of the human factor.

The next method is by using templates in Excel. This method could be used for larger datasets and is slightly less error pruning compared to the manual copy-paste method. However, it is still time consuming to create them, and for even the smallest change in the data format, the template needs to be updated. And because of the large variability of the formats, the number of templates becomes large and therefore hard to oversee, use and maintain.

Some of the researchers have learnt how to write scripts in software environments like KNIME\(^1\) and R\(^2\) to extract the data. However, this is also rather time consuming, both to learn and to write a new script for each file or dataset. Not everyone is prepared to make the effort to learn it.

One alternative is using SciStream, which is a commercial solution and an add-on to Perkin Elmer’s Spotfire suit. It helps the researcher to build a database of templates for importing single or multiple files. But this still includes a large number of manual steps by the user. And, the format of the data must conform the format given by the template. Any change in the structure of the file would demand a creation of a new template or editing the present one in order to load the data. When using SciStream for instance, the user must manually enter the number of rows to skip in file header, how many rows and columns a block of data has et cetera. This is far from the automatic process which we at CBCS require.

\(^1\)Read more about the software environment at [www.knime.org](http://www.knime.org).
\(^2\)Read more about the software environment and language at [www.r-project.org](http://www.r-project.org).
Appendix C

Input Files

Figure C.1: Screenshot of a file written by a type of machine called Envision. The machine write in the same format every time.
Table C.1: Files in dataset 1 and their respective file type. The rest of the files are in Table C.2 Dataset A is the files in dataset 1 which is of type Victor 3.

<table>
<thead>
<tr>
<th>Filename</th>
<th>File type</th>
</tr>
</thead>
<tbody>
<tr>
<td>1.2 area dual luciferase dispensor 96_20150424_840</td>
<td>Envision</td>
</tr>
<tr>
<td>1.2 area dual luciferase dispensor 96_20150424_841</td>
<td>Envision</td>
</tr>
<tr>
<td>1.2 area dual luciferase dispensor 96_20150424_842</td>
<td>Envision</td>
</tr>
<tr>
<td>1.2 area dual luciferase dispensor 96_20150508_866</td>
<td>Envision</td>
</tr>
<tr>
<td>1.2 area dual luciferase dispensor 96_20150508_867</td>
<td>Envision</td>
</tr>
<tr>
<td>0.006</td>
<td>microbeta</td>
</tr>
<tr>
<td>0.007</td>
<td>microbeta</td>
</tr>
<tr>
<td>A00_000.txt</td>
<td>microbeta</td>
</tr>
<tr>
<td>A00_001.txt</td>
<td>microbeta</td>
</tr>
<tr>
<td>A02_007.txt</td>
<td>microbeta</td>
</tr>
<tr>
<td>2000034_838.xls</td>
<td>victor2</td>
</tr>
<tr>
<td>2000034_839.xls</td>
<td>victor2</td>
</tr>
<tr>
<td>2012Amplex red ultra95020120821001.xls</td>
<td>victor2</td>
</tr>
<tr>
<td>2012Amplex red ultra95120120821002.xls</td>
<td>victor2</td>
</tr>
<tr>
<td>2012Amplex red ultra95220120821003.xls</td>
<td>victor2</td>
</tr>
<tr>
<td>2012Amplex red ultra95320120821004.xls</td>
<td>victor2</td>
</tr>
<tr>
<td>2012Amplex red ultra95420120821005.xls</td>
<td>victor2</td>
</tr>
<tr>
<td>2012Amplex red ultra95520120821006.xls</td>
<td>victor2</td>
</tr>
<tr>
<td>Abs_630_num_0.1s_3842220120530.xls</td>
<td>victor2</td>
</tr>
<tr>
<td>Abs_630_num_0.1s_3842320120530.xls</td>
<td>victor2</td>
</tr>
<tr>
<td>Abs_630_num_0.1s_3842420120621.xls</td>
<td>victor2</td>
</tr>
<tr>
<td>Abs_630_num_0.1s_3842520120624.xls</td>
<td>victor2</td>
</tr>
<tr>
<td>Abs_630_num_0.1s_3842620120624.xls</td>
<td>victor2</td>
</tr>
<tr>
<td>Abs_630_num_0.1s_3842720120627.xls</td>
<td>victor2</td>
</tr>
<tr>
<td>Abs_630_num_0.1s_3842820120627.xls</td>
<td>victor2</td>
</tr>
<tr>
<td>Filename</td>
<td>File type</td>
</tr>
<tr>
<td>----------</td>
<td>-----------</td>
</tr>
<tr>
<td>Abs_630_nm_0.1s_3842920120627.xls</td>
<td>victor2</td>
</tr>
<tr>
<td>Abs_630_nm_0.1s_3843020120627.xls</td>
<td>victor2</td>
</tr>
<tr>
<td>Abs_630_nm_0.1s_3843120120627.xls</td>
<td>victor2</td>
</tr>
<tr>
<td>Abs_630_nm_0.1s_3843220120629.xls</td>
<td>victor2</td>
</tr>
<tr>
<td>Abs_630_nm_0.1s_3843320120629.xls</td>
<td>victor2</td>
</tr>
<tr>
<td>Abs_630_nm_0.1s_3843420120702.xls</td>
<td>victor2</td>
</tr>
<tr>
<td>Abs_630_nm_0.1s_3843520120702.xls</td>
<td>victor2</td>
</tr>
<tr>
<td>Abs_630_nm_0.1s_3843620120702.xls</td>
<td>victor2</td>
</tr>
<tr>
<td>Abs_630_nm_0.1s_3843720120702.xls</td>
<td>victor2</td>
</tr>
<tr>
<td>Abs_630_nm_0.1s_3843820120702.xls</td>
<td>victor2</td>
</tr>
<tr>
<td>Abs_630_nm_0.1s_3843920120702.xls</td>
<td>victor2</td>
</tr>
<tr>
<td>Abs 405 nm kinetics 384020000371345120160315.xls</td>
<td>victor3</td>
</tr>
<tr>
<td>Abs 405 nm kinetics020000321198120150330.xls</td>
<td>victor3</td>
</tr>
<tr>
<td>Abs 405 nm kinetics020000321268320151023.xls</td>
<td>victor3</td>
</tr>
<tr>
<td>Abs 405 nm kinetics020000321297420151202.xls</td>
<td>victor3</td>
</tr>
<tr>
<td>Abs 405 nm kinetics0200001710682320140529.xls</td>
<td>victor3</td>
</tr>
<tr>
<td>Abs 405 nm0200001710692420140529.xls</td>
<td>victor3</td>
</tr>
<tr>
<td>Abs 405 nm0200001711723220141216.xls</td>
<td>victor3</td>
</tr>
<tr>
<td>Abs 405 nm0200001711733320141216.xls</td>
<td>victor3</td>
</tr>
<tr>
<td>Abs 405 nm0200001711743420141216.xls</td>
<td>victor3</td>
</tr>
<tr>
<td>Abs 405 nm0200001711763620141216.xls</td>
<td>victor3</td>
</tr>
<tr>
<td>Abs 405 nm0200001711863720150302.xls</td>
<td>victor3</td>
</tr>
<tr>
<td>Abs 405 nm0200001711873820150303.xls</td>
<td>victor3</td>
</tr>
<tr>
<td>Abs 405 nm0200001711883920150303.xls</td>
<td>victor3</td>
</tr>
<tr>
<td>Abs 405 nm0200001711894020150303.xls</td>
<td>victor3</td>
</tr>
<tr>
<td>Abs 405 nm0200001711904120150303.xls</td>
<td>victor3</td>
</tr>
</tbody>
</table>
Table C.3: Files in dataset B and their respective file type. The rest of the files are found in Table C.4.

<table>
<thead>
<tr>
<th>Filename</th>
<th>File type</th>
</tr>
</thead>
<tbody>
<tr>
<td>AMC Karin IRAP kinetics02000034121020150609.xls</td>
<td>Victor 3</td>
</tr>
<tr>
<td>AMC Karin IRAP kinetics02000034121120150609.xls</td>
<td>Victor 3</td>
</tr>
<tr>
<td>AMC fluorescence02000023124320150702.xls</td>
<td>Victor 3</td>
</tr>
<tr>
<td>AMC fluorescence02000023124520150702.xls</td>
<td>Victor 3</td>
</tr>
<tr>
<td>AMC fluorescence02000023124720150702.xls</td>
<td>Victor 3</td>
</tr>
<tr>
<td>AMC fluorescence02000023128020151125.xls</td>
<td>Victor 3</td>
</tr>
<tr>
<td>Abs 405 nm kinetics 384020000371345120160315.xls</td>
<td>Victor 3</td>
</tr>
<tr>
<td>Abs 405 nm kinetics020000321198120150330.xls</td>
<td>Victor 3</td>
</tr>
<tr>
<td>Abs 405 nm kinetics020000321268320151023.xls</td>
<td>Victor 3</td>
</tr>
<tr>
<td>Abs 405 nm kinetics020000321297420151202.xls</td>
<td>Victor 3</td>
</tr>
<tr>
<td>Abs 405 nm0200001710682320140529.xls</td>
<td>Victor 3</td>
</tr>
<tr>
<td>Abs 405 nm0200001710692420140529.xls</td>
<td>Victor 3</td>
</tr>
<tr>
<td>Abs 405 nm020000171173220141216.xls</td>
<td>Victor 3</td>
</tr>
<tr>
<td>Abs 405 nm020000171173320141216.xls</td>
<td>Victor 3</td>
</tr>
<tr>
<td>Abs 405 nm020000171174320141216.xls</td>
<td>Victor 3</td>
</tr>
<tr>
<td>Abs 405 nm020000171176320141216.xls</td>
<td>Victor 3</td>
</tr>
<tr>
<td>Abs 405 nm020000171186320150302.xls</td>
<td>Victor 3</td>
</tr>
<tr>
<td>Abs 405 nm020000171187320150303.xls</td>
<td>Victor 3</td>
</tr>
<tr>
<td>Abs 405 nm020000171188320150303.xls</td>
<td>Victor 3</td>
</tr>
<tr>
<td>Abs 405 nm020000171190420150303.xls</td>
<td>Victor 3</td>
</tr>
<tr>
<td>Abs 630 nm - 384 well - 0.1s120000220131120990.xls</td>
<td>Victor 3</td>
</tr>
<tr>
<td>Abs 630 nm - 384 well - 0.1s1200000220131119991.xls</td>
<td>Victor 3</td>
</tr>
<tr>
<td>Abs 630 nm - 384 well - 0.1s1200000220131120992.xls</td>
<td>Victor 3</td>
</tr>
<tr>
<td>Abs 630 nm - 384 well - 0.1s1200000220131120993.xls</td>
<td>Victor 3</td>
</tr>
<tr>
<td>Abs 630 nm - 384 well - 0.1s1200000220131120994.xls</td>
<td>Victor 3</td>
</tr>
<tr>
<td>Abs 630 nm - 384 well - 0.1s1200000220131120995.xls</td>
<td>Victor 3</td>
</tr>
<tr>
<td>Abs 630 nm - 384 well - 0.1s1200000220131120996.xls</td>
<td>Victor 3</td>
</tr>
<tr>
<td>Abs 630 nm - 384 well - 0.1s1200000220131120997.xls</td>
<td>Victor 3</td>
</tr>
<tr>
<td>Abs 630 nm - 384 well - 0.1s1200000220131120998.xls</td>
<td>Victor 3</td>
</tr>
<tr>
<td>Abs 630 nm - 384 well - 0.1s1200000220131120999.xls</td>
<td>Victor 3</td>
</tr>
<tr>
<td>Abs 630 nm - 384 well - 0.1s1200000220131121000.xls</td>
<td>Victor 3</td>
</tr>
<tr>
<td>Abs 630 nm - 384 well - 0.1s1200000220131121001.xls</td>
<td>Victor 3</td>
</tr>
<tr>
<td>Abs 630 nm - 384 well - 0.1s12000002201311221002.xls</td>
<td>Victor 3</td>
</tr>
<tr>
<td>Abs 630 nm - 384 well - 0.1s12000002201311221003.xls</td>
<td>Victor 3</td>
</tr>
<tr>
<td>Abs 630 nm - 384 well - 0.1s12000002201311251004.xls</td>
<td>Victor 3</td>
</tr>
<tr>
<td>Abs 630 nm - 384 well - 0.1s12000002201311251005.xls</td>
<td>Victor 3</td>
</tr>
</tbody>
</table>
Table C.4: Files in dataset B and their respective file type. The rest of the files are found in Table C.3.

<table>
<thead>
<tr>
<th>Filename</th>
<th>File type</th>
</tr>
</thead>
<tbody>
<tr>
<td>Abs 630 nm - 384 well - 0.1s12000002201311251006.xls</td>
<td>Victor 3</td>
</tr>
<tr>
<td>Abs 630 nm - 384 well - 0.1s12000002201311251007.xls</td>
<td>Victor 3</td>
</tr>
<tr>
<td>Abs 630 nm - 384 well - 0.1s12000002201311251008.xls</td>
<td>Victor 3</td>
</tr>
<tr>
<td>Abs 630 nm - 384 well - 0.1s12000002201311251009.xls</td>
<td>Victor 3</td>
</tr>
<tr>
<td>Abs 630 nm - 384 well - 0.1s12000002201311251010.xls</td>
<td>Victor 3</td>
</tr>
<tr>
<td>Abs 630 nm - 384 well - 0.1s12000002201311271011.xls</td>
<td>Victor 3</td>
</tr>
<tr>
<td>Abs 630 nm - 384 well - 0.1s12000002201311271012.xls</td>
<td>Victor 3</td>
</tr>
<tr>
<td>Abs 630 nm - 384 well - 0.1s12000002201311271013.xls</td>
<td>Victor 3</td>
</tr>
<tr>
<td>Abs 630 nm - 384 well - 0.1s12000002201311271014.xls</td>
<td>Victor 3</td>
</tr>
<tr>
<td>Abs 630 nm - 384 well - 0.1s12000002201311271015.xls</td>
<td>Victor 3</td>
</tr>
<tr>
<td>Abs 630 nm - 384 well - 0.1s12000002201312101018.xls</td>
<td>Victor 3</td>
</tr>
<tr>
<td>Abs 630 nm - 384 well - 0.1s12000002201312101019.xls</td>
<td>Victor 3</td>
</tr>
<tr>
<td>Abs 630 nm - 384 well - 0.1s12000002201312101020.xls</td>
<td>Victor 3</td>
</tr>
<tr>
<td>Abs 630 nm - 384 well - 0.1s12000002201405231052.xls</td>
<td>Victor 3</td>
</tr>
<tr>
<td>Abs 630 nm - 384 well - 0.1s12000002201512151330.xls</td>
<td>Victor 3</td>
</tr>
<tr>
<td>Abs 630 nm - 384 well - 0.1s12000002201512151331.xls</td>
<td>Victor 3</td>
</tr>
<tr>
<td>Abs 630 nm - 384 well - 0.1s12000002201512151332.xls</td>
<td>Victor 3</td>
</tr>
<tr>
<td>Abs 630 nm - 384 well - 0.1s12000002201512161333.xls</td>
<td>Victor 3</td>
</tr>
<tr>
<td>Abs 630 nm - 384 well - 0.1s12000002201512161334.xls</td>
<td>Victor 3</td>
</tr>
<tr>
<td>Abs 630 nm - 384 well - 0.1s12000002201512161335.xls</td>
<td>Victor 3</td>
</tr>
<tr>
<td>Abs 630 nm - 384 well - 0.1s12000002201512161336.xls</td>
<td>Victor 3</td>
</tr>
<tr>
<td>Abs 630 nm - 384 well - 0.1s12000002201512161337.xls</td>
<td>Victor 3</td>
</tr>
<tr>
<td>Abs 630 nm - 384 well - 0.1s12000002201512161338.xls</td>
<td>Victor 3</td>
</tr>
<tr>
<td>Abs 630 nm - 384 well - 0.1s12000002201602081340.xls</td>
<td>Victor 3</td>
</tr>
<tr>
<td>Abs 630 nm - 384 well - 0.1s12000002201602101341.xls</td>
<td>Victor 3</td>
</tr>
<tr>
<td>Abs 630 nm - 384 well - 0.1s12000002201602121342.xls</td>
<td>Victor 3</td>
</tr>
<tr>
<td>Abs 630 nm - 384 well - 0.1s12000002201602151343.xls</td>
<td>Victor 3</td>
</tr>
<tr>
<td>Abs 630 nm - 384 well - 0.1s12000002201602191344.xls</td>
<td>Victor 3</td>
</tr>
<tr>
<td>Abs 630 nm - 384 well - 0.1s12000002201603161347.xls</td>
<td>Victor 3</td>
</tr>
<tr>
<td>Abs 630 nm - 384 well - 0.1s12000002201603161348.xls</td>
<td>Victor 3</td>
</tr>
<tr>
<td>Abs 630 nm - 384 well - 0.1s12000002201603171349.xls</td>
<td>Victor 3</td>
</tr>
<tr>
<td>CellTiter-Glo protocol 96well0200000920140825_1094.xls</td>
<td>Victor 3</td>
</tr>
<tr>
<td>CellTiter-Glo protocol 96well0200000920140825_1095.xls</td>
<td>Victor 3</td>
</tr>
<tr>
<td>Course lab 131105.xls</td>
<td>Victor 3</td>
</tr>
<tr>
<td>IRAP Karin är nyfiken.xls</td>
<td>Victor 3</td>
</tr>
<tr>
<td>Karin tjuvik 160315.xls</td>
<td>Victor 3</td>
</tr>
<tr>
<td>Karin tjuvik 2 160315.xls</td>
<td>Victor 3</td>
</tr>
<tr>
<td>Nilsson_151124.xls</td>
<td>Victor 3</td>
</tr>
</tbody>
</table>
Figure C.2: Screenshot of a file written by a type of machine called Microbeta. The machine write in the same format every time.
Figure C.3: Screenshot of a file written by a type of machine called Victor 2. The machine write in the same format every time.

Figure C.4: Screenshot of a file written by a type of machine called Victor 3. The machine write in the same format every time. This file contains a 384 plate map, marked by the gridded area. The plate map does not have all its entries filled with measurement values because not all cells are filled within the gridded area.
Appendix D

List of metrics used to evaluate hypotheses

The following metrics are used to evaluate each hypothesis, together with some comments on how they could be used or why they are significant:

- **instrument**: The instrument used in the experiment to measure and printout the given file.
- **filename**: Contains information about what type of experiment etc.. None of that was used in this study.
- **clustering_method**: Clustering method used in the hypothesis generation step.
- **n_blocks**: Total number of blocks found in the hypothesis step, both number blocks, string blocks and mixed blocks. If the number of blocks as known this metric is an easy way to see which hypothesis seems to be correct.
- **n_row**: Total number of rows found in the hypothesis step. In other words, the maximum row index found plus one. This metric could be used to early screen out poorly performing hypotheses which for example discard too many elements.
- **n_col**: Total number of columns found in the hypothesis step. In other words, the maximum column index found plus one. This metric could be used to early screen out poorly performing hypotheses which for example discard too many elements.
- **n_str**: Total number of string elements found. This metric could be used when choosing clustering method.
- **n_int**: Number of integer elements found. Same usage as above.
- **n_float**: Number of float elements found. Same usage as above.
• **col of most freq min range**: The index of the column in the number blocks in which the smallest range among all values in that column was found the most times.

• **n col most freq min range**: Number of columns having the same index as the index *col of most freq min range*.

• **row of most freq min range**: The same metric as above but looking at rows instead of columns.

• **n row most freq min range**: The same metric as above but looking at rows instead of columns.

• **col of most freq max range**: The same metric as above but using largest instead of smallest range.

• **n col most freq max range**: The same metric as above but looking at rows using largest instead of smallest range.

• **row of most freq max range**: The same metric as above but looking at rows and using largest instead of smallest range.

• **n row most freq max range**: The same metric as above but looking at rows and using largest instead of smallest range.

• **most freq rowdiff**: The most frequent difference in rows between two blocks/clusters.

• **sec most freq rowdif**: The second most frequent difference in rows between two blocks/clusters.

• **third most freq rowdif**: The third most frequent difference in rows between two blocks/clusters.

• **n 96 blocks**: Number of blocks containing 96 elements.

• **n 384 block**: Number of blocks containing 384 elements.

• **n 1536 block**: Number of blocks containing 1536 elements.

• **n 96 number blocks**: Number of blocks containing 96 elements of type integer or double.

• **n 384 number blocks**: Number of blocks containing 384 elements of type integer or double.

• **n 1536 number block**: Number of blocks containing 1536 elements of type integer or double.

• **n lost element**: Number of elements discarded during hypothesis generation until after creating the nnd.

• **n elements lost before nnd**: Number of elements discarded during hypothesis generation until before creating the nnd.
• **n_blocksize**: Number of distinct block sizes.

• **most_frequent_blocksize**: The most frequent block size.

• **second_most_frequent_blocksize**: The second most frequent block size.

• **third_most_frequent_blocksize**: The third most frequent block size.

• **n_parents_in_labeltree**: Number of parents to leaf nodes in the labeltree.

The reason behind using ranges when creating metrics is because we want to find the control rows and columns in the data from a tinter plate. Remember, there are almost always one group of negative control wells and one group of positive control wells in a single plate. The controls are usually placed as columns, usually as the far right columns. But the negative and the positive control group could also be placed in the same column, one of the groups above the other one.

The purpose of a negative control well is to display the substrate when it does not respond to the experiment. The purpose of a positive control well is to display a substrate which responds fully to the experiment. Together, they are used as references to the other wells where the real experiment actually happens. For example, the negative control wells display a response of near 0 percent and and the positive control wells display a response of near 100 percent. One can then expect the other wells to display values ranging between 0 and 100 percent. These two values can also be used to normalise the other values. Outliners outside of this range could be considered to be thrown away.
Appendix E

Additional Result – The
Unsupervised Learning Approach
Figure E.1: Example of a trivial clustering result. Cells belonging to the given cluster are highlighted in green. The clustering is trivial because all cells in this sheet is clustered into the same cluster, thus all cells are highlighted. (Batch size = 60,000, number of clusters = 70, sheetname ‘Plate’ in file ‘Abs 405 nm0200001711904120150303.xls’, highlighting cluster 18.)
Figure E.2: Example of poor clustering result. Cells belonging to the given cluster are highlighted in green. The clustering is poor because cells containing numbers and cells containing strings are mixed together. (Batch size = 60 000, number of clusters = 70, sheetname 'Plate' in file 'Abs 405 nm0200001710692420140529.xls', highlighting cluster 12. The number of rows presented in the figure is limited due to limitation in how much to present on a single page. In the real file there are more rows and cells.)
Appendix F

Additional Result – The Hypothesis-generating Approach

The Figures $7.3$, $F.1$, $F.3$, $F.5$, $8.6$, $F.2$, $F.4$ and $F.6$ are all produced by clustering in the sheet called Plate in file ‘AMC fluorescence02000023124720150702.xls’, which is a part of dataset B.

![Scree plot for complete linkage](image)

**Figure F.1:** Scree plot for with complete linkage.
(a) First cut, resulting in 3 clusters.  
(b) Second cut, resulting in 6 clusters.  
Figure F.2: Complete linkage with first and second cut, resulting in 3 and 6 clusters respectively.

Figure F.3: Scree plot for weighted linkage.
(a) First cut, resulting in 3 clusters. (b) Second cut, resulting in 2 clusters.

**Figure F.4:** Weighted linkage with first and second cut, resulting in 3 and 2 clusters respectively.

**Figure F.5:** Scree plot for with average linkage.
(a) First cut, resulting in 2 clusters. 
(b) Second cut, resulting in 4 clusters.

Figure F.6: Average linkage with first and second cut, resulting in 2 and 4 clusters respectively.
Appendix G

Pre-study: Software comparison

Table G.1: Related software applications and their support for the function.

<table>
<thead>
<tr>
<th>Software</th>
<th>Description</th>
<th>Licence</th>
<th>Supported File formats</th>
<th>Tools for databloading</th>
</tr>
</thead>
<tbody>
<tr>
<td>Quva</td>
<td>“Quva Flow is an engine that consists of services running data transfer, storage, analytics, and user interface.”</td>
<td>Proprietary, closed source</td>
<td>Not found.</td>
<td>None found.</td>
</tr>
<tr>
<td>Trifacta Wrangler</td>
<td>Trifacta Wrangler Enterprise sits between the Hadoop platform, leveraged for data storage and processing; and the visualisation, analytics, or machine learning applications used downstream in the process.</td>
<td>Proprietary</td>
<td>CSV, text, Excel</td>
<td>None found.</td>
</tr>
</tbody>
</table>
| SciStream     | “SciStream, the new add-on to TIBCO Spotfire, enables scientists to import instrument output into TIBCO Spotfire easier than ever before. SciStream software facilitates quick mash up and analysis of data from multiple experiments or assays, multiple screens, and multiple instruments or detection modalities; the software allows you to compare, confirm, and uncover relationships you couldn’t before.” | Proprietary, closed source | Flow cytometer data files (FCS 3.0 and 3.1). Full support of their own hardware imaging systems. | The databoading is automated.  
  • Generic plate data import.  
  • For virtually any plate reader, scanner, or HCS reader.  
  • Multi-parametric data.  
  • Kinetic or time lapse data (label-free, calcium flux, etc).  
  • Multi-parametric and time lapse data in a single file.  
  • Data in a block format.  
  • Data with file, plate, block headers and footers.  
  • Extract important metadata and information from header and footer rows, columns, or cells.  
  • Batch import multiple plate data files into TIBCO Spotfire.  
  • Batch import High Content Screening data files (well and cell level data). |
| GNU Octave    |                                                                              | GNU GPL               | Identical to Matlab                                                                    | Similar to Matlab (limited manual tools).                                              |
| Massive Online Analysis | A free open-source software made for data stream mining with concept drift. | GNU GPL               | ArffFileStream.                                                                      | Various filters to be applied to the streamed data.                                    |

1 https://github.com/Quva/sparkplug
3 https://www.trifacta.com/products/wrangler-enterprise/
<table>
<thead>
<tr>
<th>Software</th>
<th>Description</th>
<th>Licence</th>
<th>Supported File formats</th>
<th>Tools for dataloading</th>
</tr>
</thead>
<tbody>
<tr>
<td>Maple</td>
<td>“Maple combines the world’s most powerful math engine with an interface that makes it extremely easy to analyze, explore, visualize, and solve mathematical problems.”</td>
<td>Proprietary</td>
<td>Maple's internal file format .m or .M, library archives .mla, .lib, and ind, help database file .help, Maple worksheet file .maplet and .mw. Other formats supported include: Matlab, Matrix Market, or Tab Delimited.</td>
<td>None.</td>
</tr>
<tr>
<td>IBM SPSS</td>
<td>A software package used for statistical analysis. Long produced by SPSS Inc., it was acquired by IBM in 2009. The current versions (2015) are officially named IBM SPSS Statistics. Companion products in the same family are used for survey authoring and deployment (IBM SPSS Data Collection), data mining (IBM SPSS Modeler), test analytics, and collaboration and deployment (batch and automated scoring services).</td>
<td>Proprietary</td>
<td>SPSS default file extensions associations: .chj, .cht, .clo, .chj, .cht, .clo, .caplan, .caplan, .por, .sav, .abs, .act, .ansr, .apo, .app, .apo, .spss, .apv, .clo. Common file extensions used by SPSS: .pdf, .xls. Other file extensions associated with SPSS: .lib.</td>
<td>A wizard for manually importing data from for example CSV or text files. Templates are supported.</td>
</tr>
<tr>
<td>Wolfram Language for Mathematica and Wolfram Programming Cloud</td>
<td>A symbolic mathematical computation program, sometimes called a computer algebra program, used in many scientific, engineering, mathematical, and computing fields.</td>
<td>Proprietary, closed source.</td>
<td>Hundreds of various file formats using the import command.</td>
<td>Can automatically pick manually specified columns from tables. There are also a large number of post-processing functions.</td>
</tr>
<tr>
<td>Statistica</td>
<td>An analytics software package which provides data analysis, data management, statistics, data mining, machine learning, text analytics and data visualization procedures.</td>
<td>Proprietary, closed source.</td>
<td>.pdf, .txt, .html, .xml, .doc, .rtf, proprietary STATA format, character-delimited files, Excel files, ODBC interface, SAS XPORT, Haver Analytics .dat and many more</td>
<td>Clustering techniques to identify similar documents. Ways of settings up manual controls to find malformed data. Ways to filter loaded data.</td>
</tr>
<tr>
<td>Weka</td>
<td>Weka is a collection of machine learning algorithms for data mining tasks. The algorithms can either be applied directly to a dataset or called from your own Java code. Weka contains tools for data preprocessing, classification, regression, clustering, association rules, and visualization. It is also well-suited for developing new machine learning schemes.</td>
<td>open source software issued under the GNU GPL.</td>
<td>ARFF (Attribute-Relation File Format) (default file format). XRFF (eXtensible attribute-Relation File Format).  Weka automatically recognizes a file being gzip compressed, if the file's extension is .xrff.gz instead of .xrff. CSV (with some drawbacks: UTF-8 datasets/files need to explicitly set the file encoding to utf-8). Support to convert CSV to ARFF. ARFF to database importer.</td>
<td></td>
</tr>
<tr>
<td>Software</td>
<td>Description</td>
<td>Licence</td>
<td>Supported File formats</td>
<td>Tools for dataloading</td>
</tr>
<tr>
<td>----------------</td>
<td>------------------------------------------------------------------------------</td>
<td>-------------</td>
<td>---------------------------------------------------------------------------------------</td>
<td>----------------------------------------------</td>
</tr>
<tr>
<td>RapidMiner</td>
<td>provides an integrated environment for machine learning, data mining, text mining, predictive analytics and business analytics. It is used for business and industrial applications as well as for research, education, training, rapid prototyping, and application development and supports all steps of the data mining process including results visualization, validation, and optimization. Previous version is available under an OSI-certified open source license (AGPL).</td>
<td>Proprietary</td>
<td>.sav, .arff, .mdf, .sas, .excel-file (with our without the format information about the cells), .xml, .txt. .BibTexX-files</td>
<td>None found.</td>
</tr>
<tr>
<td>UNISTAT</td>
<td>The Unistat computer program is a statistical data analysis tool featuring two modes of operation. The stand-alone user interface is a complete workplace for data input, analysis and visualization while the Microsoft Excel add-in mode extends the features of the mainstream spreadsheet application with powerful analytical capabilities.</td>
<td>Proprietary</td>
<td>All formats supported by Microsoft Excel: ODBC database support, UNISTAT-specific format (USW), .html, formats created by dBase, Data Processor, Lotus, Text Delimited, Text Free, Text Fixed, SQL, DIF, Access, FoxPro, Paradox</td>
<td>Automatically detects file format (of the supported formats) based on the first few bytes of a file. If the file cannot be detected, it's contents (read as text) is shown to the user, which is then prompted to manually select a delimiter in order to parse the data as a 2d data matrix.</td>
</tr>
<tr>
<td>SAS Institute</td>
<td>SAS stands for the Statistical Analysis System, a software system for data analysis and report writing. SAS is a group of computer programs that work together to store data values and retrieve them, modify data, compute simple and complex statistical analyses, and create reports. Proprietary.</td>
<td>Proprietary</td>
<td>.sas, .as2, .as7, .log, .a7, .a2, .as7, .asm2, .sm7, .as17, .as2, .as7, .asm2, .sm7, .as17, .as2, .as7, .asm2, .sm7, .as17, .as2, .as7, .asm2, .sm7, .as17, .as2, .as7, .asm2, .sm7, .as17, .as2, .as7, .asm2, .sm7, .as17, .as2, .as7, .asm2, .sm7, .as17, .as2, .as7, .asm2, .sm7, .as17, .as2, .as7, .asm2</td>
<td>A simple wizard or scripts for importing.</td>
</tr>
<tr>
<td>DADiSP</td>
<td>“A modular scientific software framework. It provides all the functionalities needed to deal with big data processing, statistical analysis, visualisation and storage. It is mainly written in C++ but integrated with other languages such as Python and R.”</td>
<td>Proprietary</td>
<td>.cav, .mat, .mat. Plugins to add support for ISF, SIF, PNRF, UFF, WVF, Datap files.</td>
<td>Limited manual control of loading tabular text data through its READTABLE function.</td>
</tr>
<tr>
<td>ROOT Data Analysis Framework</td>
<td>Provides an open-source scientific software framework that is used to store code and graphics. It is made to be extendible, but has no special built-in support for anything else. Proprietary.</td>
<td>LGPL</td>
<td>Custom ROOT file structure is used to store code and graphics.</td>
<td>None.</td>
</tr>
<tr>
<td>icCube</td>
<td>an in-memory multidimensional online analytical processing (OLAP) server.</td>
<td>Proprietary</td>
<td>.cav, .excel.xls and .xlsx, XML for Analysis XMLA, Google Bigtable databases, MongoDB databases and plugins to load more types.</td>
<td>Various ways of manually filtering data.</td>
</tr>
<tr>
<td>Talend Open Studio</td>
<td>Talend Open studio is an open source software by Talend. Talend (Pronunciation: TAL-end) is a software vendor specializing in Big Data Integration. The company provides big data, cloud, data integration, data management, master data management, data quality and enterprise application integration technology and services. Open source (Talend Open studio, other versions are proprietary.)</td>
<td>Proprietary</td>
<td>a large variety of file formats, including XML, Excel, LDIF, EBCDIC, archive files, and generic delimited text files.</td>
<td>Wizard that helps you to manually load data.</td>
</tr>
<tr>
<td>Dataconvert</td>
<td>Unified python library and command line interface to convert data from one format to another (especially tabular data). Can be used together with Dataproxy, a web application that can convert between tabular data.</td>
<td>MIT license</td>
<td>.cav (to, from, including type detection of dates/numbers etc.), .xls/.xlsx (from), .json (T0), .XML to GeoJSON</td>
<td>None found.</td>
</tr>
</tbody>
</table>

7. https://root.cern.ch/
### Table G.1: Related software applications and their support for the function.

<table>
<thead>
<tr>
<th>Software</th>
<th>Description</th>
<th>Licence</th>
<th>Supported File formats</th>
<th>Tools for dataloading</th>
</tr>
</thead>
<tbody>
<tr>
<td>CDD Vault - Drug Discovery</td>
<td>CDD Vault is a hosted database solution for secure management, analysis, and sharing of chemical and biological data. It lets you intuitively organize chemical structures and biological study data, and collaborate with internal or external partners through an easy to use web interface. Drug discovery-teams (Yiva) can also donate data to the database.</td>
<td>proprietary, closed source.</td>
<td>Data Import Wizard offers reusable templates. Upload from Excel, SD, SMILES (The Simplified Molecular Input Line Entry Specification is a line notation for molecules).</td>
<td>New API facilitates bi-directional integration with existing resources. A data Import Wizard and reusable templates streamline migration. Manually choose size of microtiter plate, position of positive and negative control.</td>
</tr>
<tr>
<td>Harvard Screen-saver</td>
<td></td>
<td>GNU GPL</td>
<td>Files are loaded from a database. Hardcoded plates.</td>
<td>None.</td>
</tr>
<tr>
<td>ELKI</td>
<td></td>
<td>AGPLv3</td>
<td>ELKI-specific text based format also supported by GnuPlot, similar to space-separated values. CSV, ARFF, LibSVM. Comes with methods to help implement your own parser.</td>
<td>None.</td>
</tr>
<tr>
<td>Julia</td>
<td>Julia is a high-level, high-performance dynamic programming language for technical computing, with syntax that is familiar to users of other technical computing environments.</td>
<td>MIT License, Open Source</td>
<td>Any (including ODBC), parsing is limited to character-delimited files.</td>
<td>Julia's readslim function can be used with table data to automatically pick column headers and to manually skip the first few lines of input.</td>
</tr>
<tr>
<td>InfoZoom</td>
<td>Data analysis, business intelligence and data visualization software product created using in-memory analytics.</td>
<td>Closed source.</td>
<td>• InfoZoom files (.fox, .foc or .fop).</td>
<td>The InfoZoom import/export file contains no information about the format of the contained values. InfoZoom replaces this information using a hypothesis-based process during import: On reading, InfoZoom checks whether all values of an attribute have the same format. If all values in the same format can be successfully interpreted, InfoZoom adopts this as attribute format. This involves a rough categorization of the attribute values as (text) string, date or numeric values. Numeric and date values are then compared with a series of patterns to determine the format details. As long as you consistently use one of those formats for all of an attribute’s values, InfoZoom identifies the format reliably. When inconsistencies arise, the user must intervene and select a Data format. Syntax for type formats.</td>
</tr>
<tr>
<td>Analytics</td>
<td>A software package for creating, analyzing and communicating quantitative decision models</td>
<td>Proprietary, closed source</td>
<td>Excel .xls, .xlsx files, ODBC support.</td>
<td>None.</td>
</tr>
<tr>
<td>PAW</td>
<td>PAW is conceived as an instrument to assist physicists in the analysis and presentation of their data. It provides interactive graphical presentation and statistical or mathematical analysis, working on objects familiar to physicists like histograms, event files (NTuples), vectors, etc. PAW is based on several components of the CERN Program Library.</td>
<td>Like CERN Program Library: PAW usage and/or redistribution is granted under the terms of the GNU General Public License.</td>
<td>.data</td>
<td>None found.</td>
</tr>
</tbody>
</table>

---

2. [https://www.collaborativedrug.com/pages/product.info](https://www.collaborativedrug.com/pages/product.info)
3. [core/arc/main/java/edu/harvard/med/screen saver/model/libraries/PlateType.java:40](http://www.ncbi.nlm.nih.gov/002767510750636/supplement.pdf)
<table>
<thead>
<tr>
<th>Software</th>
<th>Description</th>
<th>Licence</th>
<th>Supported File formats</th>
<th>Tools for dataloading</th>
</tr>
</thead>
<tbody>
<tr>
<td>Knime</td>
<td>An open source data analytics, reporting and integration platform</td>
<td>GNU GPL</td>
<td>Proprietary, knime format, R models, .arff, .csv, Excel .xls, json, xml, PMML, ODBC interface.</td>
<td>A large number of functions to manually manipulating and filtering table data. Built-in support for compressed data and data read from http or ftp.</td>
</tr>
<tr>
<td>HippoDraw</td>
<td>An object oriented statistical data analysis package.</td>
<td>GNU GPLv2</td>
<td>XML, FITS, CERN ROOT files, plain text files. Support for HDF5 via it’s Python bindings.</td>
<td>None.</td>
</tr>
<tr>
<td>Data Applied</td>
<td>Data Applied implements a collection of visualization tools and algorithms for data analysis and data mining. The product supports several types of analytical tasks, including visual reporting, tree maps, time series forecasting, correlation analysis, outlier detection, decision trees, association rules, clustering, and self-organizing maps.</td>
<td>Proprietary, closed source.</td>
<td>.csv, excel-files, base64-encoded bytes</td>
<td>None found.</td>
</tr>
<tr>
<td>MScreen</td>
<td>MScreen is an open source High Throughput Screening (HTS) data storage and analysis system that has been developed at the Center for Chemical Genomics (CCG), a core facility at the University of Michigan. MScreen processes HTS data through a web-based interface to help screeners visualize their data and simplify the hit-to-load decision process. Thomas Lundbäck knows a researcher in Umeå who uses this software. Through him we might be able to gain access.</td>
<td>Open Source</td>
<td>None found.</td>
<td>Converts the data outputs from generic formats, verifies the data, maps the data to registered sort plate libraries, and imports the data, during which parameters are assigned as statistics are calculated.</td>
</tr>
<tr>
<td>SAP Lumira</td>
<td>Data visualization software that makes it easy to create beautiful and interactive maps, charts, and infographics. Import data from Excel and many other sources, perform visual BI analysis using intuitive dashboards, and securely share insights and data stories with your team.</td>
<td>Proprietary, closed source.</td>
<td>Microsoft Excel, Microsoft Access files, .csv, data can be read from a SQL database, Hadoop HDFS, proprietary SAP BusinessObjects universe files.</td>
<td>The Data Transfer Workbench provides the following functions: Administration and organization of data transfer projects. Tools for analyzing the required SAP structures Integration of standard data transfer programs Registration and integration of your own data transfer programs and help programs Various techniques for loading data. The data is read from a transfer file in SAP format and loaded into your system using one of the techniques below: BAPI interface: API methods of SAP business object types. Batch input Direct input (DINP) Data conversion using the Legacy System Migration Workbench (LSMW).</td>
</tr>
<tr>
<td>Waffles</td>
<td>waffles is a tool that allows users to generate commands by stepping through a menu system.</td>
<td>CC0</td>
<td>ARFF, CSV</td>
<td>waffles_wizard: A graphical tool that allows users to generate commands by stepping through a menu system. waffles_transform: “tools to drop columns, swap columns, fill in missing values, sort in a particular column, shuffle rows, and numerous other useful transformations”. waffles.sparse: a tool that assigns an attribute to each possible word and row in each document, saving it as a sparse matrix.</td>
</tr>
</tbody>
</table>

1. [http://mscreen.lsi.umich.edu/](http://mscreen.lsi.umich.edu/)
3. [http://mscreen.lsi.umich.edu/mscreenwiki/videos/mscreen/MScreen_Overview.html](http://mscreen.lsi.umich.edu/mscreenwiki/videos/mscreen/MScreen_Overview.html)
4. [http://help.sap.com/saphelp_nw70/helpdata/en/0d/e211c5543e11d1895d0000e829fbbd/content.htm](http://help.sap.com/saphelp_nw70/helpdata/en/0d/e211c5543e11d1895d0000e829fbbd/content.htm)
5. [https://creativecommons.org/publicdomain/zero/1.0/](https://creativecommons.org/publicdomain/zero/1.0/)
6. [http://uaf46365.ddns.uark.edu/waffles/command/transform.html](http://uaf46365.ddns.uark.edu/waffles/command/transform.html)
<table>
<thead>
<tr>
<th>Software</th>
<th>Description</th>
<th>Licence</th>
<th>Supported File formats</th>
<th>Tools for dataloading</th>
</tr>
</thead>
<tbody>
<tr>
<td>TinkerPlots</td>
<td>A data visualization and modeling tool developed for use by middle school students through to university students. TinkerPlots can be used to teach grades 4 and up in subjects including math, statistics, social science, or physical or biological science content - in any course in which data are relevant. Stable release: September 2015.</td>
<td>Proprietary</td>
<td>Text files. Also support for loading data from internet.</td>
<td>No.</td>
</tr>
<tr>
<td>Encog</td>
<td>Encog is a machine learning framework available for Java, .Net, and C++. It supports multiple machine learning algorithms and can make use of the GPU for calculations</td>
<td>Apache 2.0 License</td>
<td>Custom Encog file formats to store training data and neural networks, very similar to CSV. Extendible with the file formats supported by the host language.</td>
<td>None.</td>
</tr>
<tr>
<td>ILNumerics</td>
<td>Proprietary computing engine written in C#</td>
<td>Proprietary, closed source.</td>
<td>Any formats supported by .NET applicable to array data. Matlab .mat files.</td>
<td>Limited manual control of loading tabular text data through its coverread function.</td>
</tr>
<tr>
<td>SciLab</td>
<td>SciLab is free and open source software for numerical computation providing a powerful computing environment for engineering and scientific applications. SciLab includes hundreds of mathematical functions. It has a high level programming language allowing access to advanced data structures, 2-D and 3-D graphical functions.</td>
<td>CeCILL (GPL-compatible). Open source</td>
<td>SciLab can read data from Excel and write data in Excel spreadsheets thanks to a dedicated external module. An Excel link is available via ATOMS. Text files encoded as ANSI/ASCII and UTF-8, with either tab or space separator (not both at the same time). CSV files: an .asci file with comma separated values delimited fields. .sci -¿ functions, or collection of functions in one file. In old versions the only way to process this file was getf('file.sci') but now you can also issue an exec('file.sci'), .sce -¿ a file supposed to be processed with exec('file.sce') mixing functions (preferably at the beginning of the file) and 'main program' instructions. Fortran binary files (4 byte or 8 byte words) not found any.</td>
<td></td>
</tr>
<tr>
<td>labVIEW</td>
<td>A system-design platform and development environment for a visual programming language</td>
<td>Proprietary, closed source.</td>
<td>Text files, .csv, proprietary binary labVIEW file, XML</td>
<td>labVIEW can automatically detect type information from text table files.</td>
</tr>
<tr>
<td>Genedata Screener</td>
<td>Genedata Screener is a comprehensive software solution that captures, analyzes, and manages all types of screening data. Designed to import data from any screening instrument, ranging from low to high throughput across the screening workflow, and used by leading pharmaceutical companies, CROs, and academic research institutions all over the world, Genedata Screener provides a single platform for all screening knowledge.</td>
<td>Closed source, proprietary.</td>
<td>Unknown</td>
<td>Import APIs based on Python or Java, maintained by the customer.</td>
</tr>
<tr>
<td>ActivityBase</td>
<td>proprietary, closed source.</td>
<td>None found.</td>
<td>None found.</td>
<td>None found.</td>
</tr>
</tbody>
</table>

Notes:
- [http://www.tinkerplots.com/overview](http://www.tinkerplots.com/overview)
- [http://www.scilab.org/scilab/about](http://www.scilab.org/scilab/about)
- [http://atoms.scilab.org/toolboxes/xls_link/](http://atoms.scilab.org/toolboxes/xls_link/)
- [https://www.genedata.com/products/screener/](https://www.genedata.com/products/screener/)
<table>
<thead>
<tr>
<th>Software</th>
<th>Description</th>
<th>Licence</th>
<th>Supported File formats</th>
<th>Tools for dataloading</th>
</tr>
</thead>
<tbody>
<tr>
<td>Matlab</td>
<td>Proprietary. Closed source.</td>
<td>Character-delimited files and MAT-files (Matlab’s own array data format). ASCII and Unicode support. Data formats that are used internally: Name: MAT-File Binary/text based. Binary file size limit: 2 GB (stored in main memory). Unicode: Yes. Compression: Yes. Other: Matrices, automatically stored as either full or sparse matrices. The latest version of the MAT-File (7.3) uses a HDF5-based format, with data loaded in multiple small chunks.</td>
<td>Detects files based on file extension. Error correction: No built-in error correction. Can load columns with a regexp (e.g. load('topo.mat', '-regexp', '.*') only loads variable names containing at least one digit). Has multiple built-in functions to help with identifying input type (e.g. isnumeric, istaltime, isduration).</td>
<td></td>
</tr>
<tr>
<td>OpenRefine</td>
<td>BSD license. Open source.</td>
<td>TSV, CSV, or values separated by a custom separator you specify. Excel (.xls, .xlsx), XML, RDF as XML, JSON, Google Spreadsheets, RDF N3 triples, zip, tar.gz, tgz, .tar.bz2, .gz, or .bz2. You can also point OpenRefine at a URL to a data file or a Google Spreadsheet. Once imported, the data is stored in OpenRefine’s own format, and your original data file is left undisturbed.</td>
<td>The user needs to manually state the delimiter, number of header lines etc.</td>
<td></td>
</tr>
<tr>
<td>cavkit</td>
<td>CSVkit is works on Python versions 2.6, 2.7, 3.3 and 3.4, as well as PyPy. It is supported on OSX and Linux. It also works—but is tested less frequently—on Windows.</td>
<td>The MIT License. Open source.</td>
<td>None found.</td>
<td></td>
</tr>
<tr>
<td>FreeMat</td>
<td>GNU GPL</td>
<td>Similar to Matlab.</td>
<td>Similar to Matlab (limited manual tools).</td>
<td></td>
</tr>
<tr>
<td>R</td>
<td>R is a programming language and software environment for statistical computing and graphics supported by the R Foundation for Statistical Computing. The R language is widely used among statisticians and data miners for developing statistical software and data analysis.</td>
<td>GNU GPL</td>
<td>txt (including Unicode), fixed width text files, Data Interchange Format (.dif), xml, csv, xls, facilities to import from files created by DBase, EpInfo, Minitab, R-PLUS, SAS, SSPS, Stata, Systat, Octave. Bindings for a large number of databases (including MySQL, PostgreSQL, MongoDB, Access. Full ODBC support). Packages to handle hdf5. Image formats: pixmap, bmp, tiff, packages to handle many other image types. Can read directly from BSD sockets.</td>
<td>Trivial functions to help reshape data. Multiple packages to handle data reshaping, including plyr which supports advanced reshaping of data, including data recovery.</td>
</tr>
<tr>
<td>NumPy</td>
<td>BSD 3</td>
<td>Custom binary .npy file format (which includes memory-mapping), functions to load arbitrary text files.</td>
<td>A large number of functions to help with manually loading complicated data, such as using regular expressions to parse certain fields in a character-delimited file.</td>
<td>CSVKit.</td>
</tr>
<tr>
<td>Software</td>
<td>Description</td>
<td>Licence</td>
<td>Supported File formats</td>
<td>Tools for dataloading</td>
</tr>
<tr>
<td>-------------------</td>
<td>---------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------</td>
<td>-----------------</td>
<td>-----------------------------------------------</td>
<td>-----------------------</td>
</tr>
<tr>
<td>ChemAxon - Instant JChem</td>
<td>Instant JChem is a desktop application designed for the chemist. With Instant JChem you can create, explore and share chemical and non-chemical data in local and remote databases without additional administration. Instant JChem has a wide and growing range of functionality including customisable database views, integration of library enumeration and dynamic population of columns with singular and combined molecular property and descriptor predictions.</td>
<td>Proprietary</td>
<td>A number of formats</td>
<td>File Import wizard.</td>
</tr>
<tr>
<td>SCaVis</td>
<td>SCaVis is an environment for scientific computation, data analysis and data visualization for scientists, engineers and students. The program is fully multi-platform (100% Java) and integrated with Java and a number of scripting languages: Jython (Python), Groovy, JRuby, BeanShell. SCaVis can be used to plot functions and data in 2D and 3D, perform statistical tests, data mining, numeric computations, function minimization, linear algebra, solving systems of linear and differential equations. The development of SCaVis was discontinued in 2015. Please use DMelt instead. The DMelt program is backward compatible with SCaVis.</td>
<td>Free to download. Open source</td>
<td>The official documentation is closed.</td>
<td>No documentation.</td>
</tr>
</tbody>
</table>

---

*https://www.chemaxon.com/products/instant-jchem-suite/instant-jchem/*

*https://docs.chemaxon.com/display/instantjchem/Using+File+Import*

*http://sourceforge.net/projects/scavis/*

*http://jwork.org/scavis/*

*https://fossies.org/diffs/scavis/2.1_vs_2.2/*
<table>
<thead>
<tr>
<th>Software</th>
<th>Description</th>
<th>Licence</th>
<th>Supported File formats</th>
<th>Tools for data loading</th>
</tr>
</thead>
</table>
| DataMelt    | DataMelt (or, in short, DMelt) a computation and visualization environment, is an interactive framework for scientific computation, data analysis and data visualization designed for scientists, engineers and students. | Mixed: The core engine is GPL. Also available under LGPL for commercial usage. | • Includes the native Java and Python methods for file input and outputs.  
• HFile (File format for hbase. A file of sorted key/value pairs. Both keys and values are byte arrays.) The HFile format based on Java serialization. Optionally, compression and XML serialization are supported. Data can be written sequentially or using the key-value maps.  
• The PFile format based on the Protocol Buffers engine for multi platform input output  
• The HBook format, which is a simplified XML format to write large data structures without XML tags  
• Arbitrary data structure can be written into object databases with file system as back-end. This allows writing large data collections to files which normally do not fit into the computer memory.  
• Several SQL database engines are included as external packages  
• AIDA (computing) file format (read only)  
• ROOT file format (read only)

64 [http://jwork.org/dmelt/](http://jwork.org/dmelt/)  
67 [http://www.kovcomp.co.uk/mvsp3man.pdf](http://www.kovcomp.co.uk/mvsp3man.pdf)  
| DevInfo     | A common database platform to track human development. A tool for organizing, storing and presenting data in a uniform way to facilitate data sharing.       | Database is freely available at their website. | CSV, DevInfo Data Entry, SDMX-ML.                                                        | "Technical staff" has the responsibility to manually check their colleagues when data is manually uploaded. |  |
| QDA Miner   | Proprietary. Closed source.                                                                                                                     | QDA Miner project files (.wpj),  
• Simstat data files (.dbf),  
• .csv, .pdf, DBase files,  
• Reference Information System (.ris),  
• Paradox data files and formats from a large number of word processors and spreadsheet applications.  
• Graphics formats: bmp, jpg, gif, png, wmf.  | Tools to merge files  
• Document conversion wizard which allows simple modification of data  
• Tools to filter data |  |
<table>
<thead>
<tr>
<th>Software</th>
<th>Description</th>
<th>Licence</th>
<th>Supported File formats</th>
<th>Tools for dataloading</th>
</tr>
</thead>
</table>
| Tanagra                      | Free DATA MINING software for academic and research purposes. It proposes several data mining methods from exploratory data analysis, statistical learning, machine learning and databases area. | Freeware, Open source (custom license) | • Tanagra-specific binary/ textual stream diagram (.bdm, .tdm)  
  • .csv, tab-separated text files.  
  • Can also output html files. | Trivial filtering tools when loading data                                                                                                           |
| IBM i2 Analyst’s Notebook    | Proprietary, closed source.                                                   | Analyst’s Notebook Exchange file, Cellebrite UFED XML file, MSAB XRY XML file, Excel .xls and .xlsx, .csv. | Can save “import specifications” which specifies how a certain type of data should be loaded. Will inspect data in order to automatically detect an import specificaation. |
| Trifacta Wangler             | Initially developed by Stanford Visualization Group in 2011-2013. Today it is commercial. Wrangler is an interactive tool for data cleaning and transformation. Trifacta Wrangler Enterprise sits between the Hadoop platform, leveraged for data storage and processing; and the visualisation, analytics, or machine learning applications used downstream in the process. | Proprietary                      | CVS, text, exc.                                                                                                                                      | None found.                                                                            |
| Pandas                       | A Python library for data manipulation and analysis. In particular, it offers data structures and operations for manipulating numerical tables and time series. | GNU GPL                         | Character-delimited files, .json, Excel .xls, .xlsx, HDF5, sql, .json, can connect to sql databases.                                               | Multiple ways of manually loading tabular data (such as specifying the data type of certain fields). Supports compressed text table files. |
| Data Integration - Kettle    | Data Integration (or Kettle) delivers powerful Extraction, Transformation, and Loading (ETL) capabilities, using a groundbreaking, metadata-driven approach. With an intuitive, graphical, drag and drop design environment and a proven, scalable, standards-based architecture. Data Integration is increasingly the choice for organisations over traditional, proprietary ETL or data integration tools. | Apache license 2.0, open source | text files etc.                                                                                                                                     | The Import tool provides a command line interface for the import of transformations and jobs into a repository. Manually specify what delimiter and enclosure to use. |

73 General information: http://vis.stanford.edu/wrangler/  
Hands-on Web app: http://vis.stanford.edu/wrangler/app/  
74 https://www.trifacta.com/products/wrangler-enterprise/  
75 https://community.pentaho.com/projects/data-integration/  
76 https://github.com/pentaho/pentaho-kettle/  
77 http://wiki.pentaho.com/display/EAI/Text+File+Input  
78 http://wiki.pentaho.com/display/EAI/Import+User+Documentation