Chapter 4

Stage 1: Data Extraction

4.1 GUERRILLA ANALYTICS WORKFLOW

Data Extraction is the first stage in the Guerrilla Analytics workflow (Section 2.1), as illustrated in Figure 9. It involves taking data out of some system or location so it can be brought into the analytics team's Data Manipulation Environment (DME). The place the data is extracted from is called the "source" and the environment it is brought into is called the "target." Data extraction also involves validating that the extracted data is correct and complete.

Data extraction is quite a broad set of activities. These are the main types of data extraction to consider:

- **System data extractions:** These involve connecting to the back-end database that underlies some application and taking a copy of the data in the application at a point in time. Some systems will provide a mechanism to dump data out of the system consistently. Other systems will require custom program code to query data out of the system. Back-end data extractions are desirable in a project because they provide data in its most raw form, and therefore in a form that is most flexible for the analytics team. The data has not yet been presented in the application layer, where it is often modified for user convenience. However, in this type of data extraction you will typically encounter a variety of database systems, each of which will have their own flavors of programming languages and their own custom data structures.
- **Front-end data extractions:** These involve connecting to the application front-end as a user and capturing some data that the application presents. This data is typically in the form of business intelligence reports. Some applications conveniently provide a method to export and save reports in a variety of file formats such as PDF and spreadsheets. Some applications allow you to construct custom reports by graphically arranging reporting elements. Other web applications may have to be "scraped" to grab the data that is presented in the application. Scraping is the process of using an automated method to simulate human interaction with a web page, capturing the web page information that is presented by the application. A web scraper could, for example, automatically click through all train timetables presented on a company's website and scrape those timetables into a series of datasets.

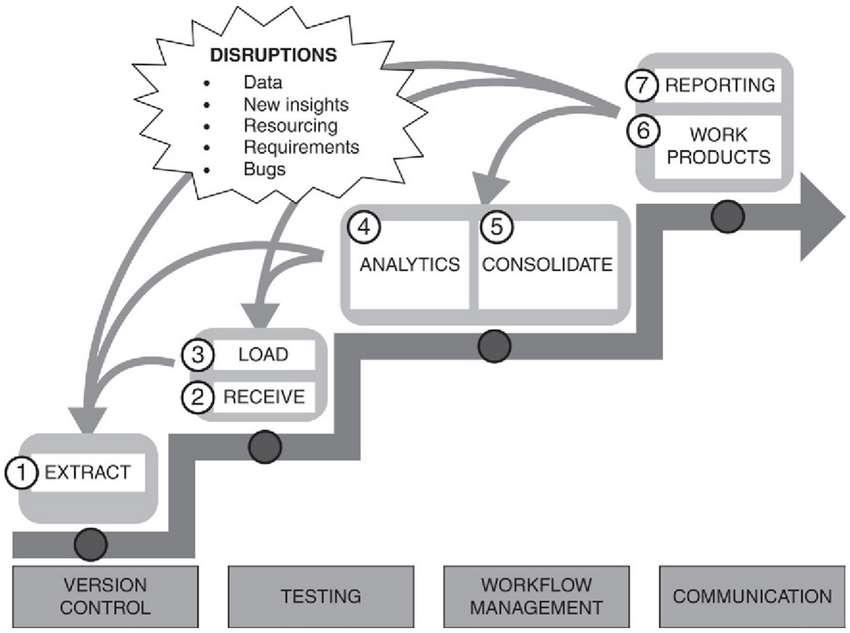


FIGURE 9 The Guerrilla Analytics workflow

- **Optical character recognition (OCR):** Despite the fact that we now live in a "data" age, there are still many occasions on which data is trapped in scanned documents. That is, printed, typed, or even hand-written documents have been scanned into an electronic file format. This is still common with legal documentation and in legacy storage in financial institutions. OCR is an automated process for turning these scanned documents into "real" digital content such as text and numbers.

The Extraction stage of the Guerrilla Analytics workflow is sometimes done by the customer's IT team as they prepare data for your team. Nonetheless, the Guerrilla Analytics team does on occasion perform data extractions themselves because of the customers' lack of capability. Either way, it is important to understand the processes involved so that data provenance can be preserved at this critical first stage in the analytics workflow.

4.2 PITFALLS AND RISKS

There are several pitfalls and risks to be aware of when extracting data. These depend on the type of data extraction that is taking place.

- **Extracting from a moving target:** The source system is not frozen before an extraction takes place. This means that data continues to be generated as the data extraction is taking place. Checks performed on the extracted data cannot be easily compared to data in the originating source system because the data has "moved on" in the intervening time since the extraction.

- **Inability to assess completeness:** The data is not quantified before extraction. When the data extraction is finished there is nothing to compare the extracted data to, and so there can be no confidence that the extraction is complete and correct.
- **No reproducibility of extraction process:** The scripts to execute the extraction (either custom scripts or commands to the system's data export utility) are not saved properly. If questions arise regarding the extracted data or the extraction needs to be repeated for a data refresh, there is no record of how the data was previously extracted.
- **Nothing to compare to:** Front-end reports are not captured from the system at the time of data extraction. Front-end reports are a useful check for the correctness of the extracted data. They are also helpful when reverse engineering a system and its data relationships, as they effectively present business rules to the user. Both of these advantages can be lost if the captured front-end reports are not in sync with the extracted data.
- **Front-end reports cannot be reproduced:** Many applications are complex. It may be possible to generate a "single" report using a large combination of options and filters. The available options and filters can also depend on the type of user logged into the system. For example, administrators will have more visibility of the system than general users. If front-end reports cannot be reproduced then they are less useful in reverse engineering the system.
- **No record of web pages:** Web pages are dynamic content. Today's *BBC* news page is different from yesterday's. If no raw copy of an extracted web page is saved then it is impossible to check if a web scrape was done correctly.
- **No checks of OCR data:** OCR is prone to errors. A number 3 often gets confused with a number 8, for example. These errors can have serious implications in financial data where numbers are very important! Thought needs to be given as to how OCR documents can be checked for consistency and correctness, especially when OCR data is not in a neat tabular form.

These pitfalls and risks are of concern to the Guerrilla Analytics team because they can invalidate all subsequent work. Under tight timelines, the team cannot afford to make these basic mistakes in acquiring data. The following tips will now guide you in ways to mitigate the effects of these data extraction risks and pitfalls.

4.3 PRACTICE TIP 1: FREEZE THE SOURCE SYSTEM DURING DATA EXTRACTION

4.3.1 Guerrilla Analytics Environment

If you wish to validate that your extracted data are correct and complete, then you need to first determine what correct and complete is. When a system is running and live, then you have a moving target. By the time you have run a data extraction, the data in the system will be newer and different to what were extracted. This leaves you with no source record for meaningful comparison.

4.3.2 Guerrilla Analytics Approach

When a data extraction is being executed, freeze the source system or clone it for the duration of the extraction.

4.3.3 Advantages

With a data extraction from a frozen or cloned system:

- **No moving target:** There is a static copy of the data from which key data characteristics and profiles can be recorded. This addresses Guerrilla Analytics concerns around data provenance.
- **Reports and data in sync:** Because the data is static, so too are the system's reports. It is therefore possible to capture useful front-end reports that are guaranteed to tie out to the underlying system data. This helps begin the process of understanding the data without the need for detailed documentation that can be difficult to acquire.

4.4 PRACTICE TIP 2: EXTRACT DATA INTO AN AGREED FILE FORMAT

4.4.1 Guerrilla Analytics Environment

Extracted data will be loaded into the analytics team's target DME. If the customer's source system and the analytics team's target system are the same, then data transfer is usually feasible using a data export format that the source system provides. Most relational databases, for example, can export their data into binary or text file format. However, it is far more common for the source and target systems to differ. This raises the question of how two different systems can exchange data safely.

Here are some examples of the thorny issues that arise. Consider the date "Tuesday 17th March 2002." Do we extract that as "Tue 17/03/2002" or "2002-03-17"? What about "03-17-02"? In terms of the extracted file format, do we assume that the first row of the file describes data field names, or are those names provided separately? Which characters separate data fields from one another? How are separator characters handled when they appear within a data field? How do we terminate an extracted file so that we know it was not truncated during extraction? There is much to consider and agree on at the outset of data extraction.

4.4.2 Guerrilla Analytics Approach

It is imperative to specify and agree the data format when extracting and exchanging data. The data format covers both the overall format of the file and the format of data fields within the file. There is no right or wrong answer here (although some answers are far better than others). What is important is that the file format is agreed on and is consistent. Some important data field format considerations are:

- What is the agreed date and date time format?
- What is the agreed floating point number format?
- What decimal separators are used, if at all? Some countries use 20,000 while others write 20.000.

Once the data field formats are agreed, then a data file format must be agreed. Perhaps the most common flat text file format in use is the Comma Separated Value (CSV) file format. Surprisingly, there is no agreed standard for a CSV file, so make sure you are clear on exactly how the CSV file is constructed. In terms of the CSV file format, consider:

- **Field separators:** The character or characters that separate fields.
- **Enclosing characters:** The character or characters that enclose fields.
- **Escaping:** How to handle the situation when a separator or enclosing character appears within a field as valid field content.

There are several attempts to agree to a CSV standard and these are useful for further reading (Shafranovich, 2005). Increasingly other self-describing formats are also in use such as XML and JSON. Again there is no right or wrong answer in the choice of formats. What is important is agreement, consistency, and awareness of the risks that can arise.

4.4.3 Advantages

Agreeing and specifying the data extraction format is critical to successful data extraction and transfer to target. Any mistake at this point affects all subsequent work with the data.

- **Reduced risk of corruption:** If field formats are not clearly agreed, data can easily be corrupted at import into the DME. A classic and all too common example is that of dates. The date 4 September (04-09-2010 in one format) becomes 9 April when interpreted as Month-Day format.
- **Ease of interpretation:** Interpretation of the file format is straightforward and unambiguous.
- **Source and target compatibility:** A file format that best suits both the source and target DME system can be agreed on to make data exchange less painful for both parties.

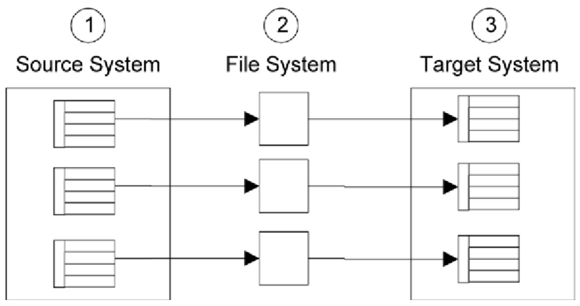


FIGURE 10 **The journey of extracted data**

4.5 PRACTICE TIP 3: CALCULATE CHECKSUMS BEFORE DATA EXTRACTION

4.5.1 Guerrilla Analytics Environment

If you think about the data's journey, there are three stages it goes through. These are illustrated in Figure 10.

- **Source:** Dataset on the original system, which may have its own proprietary data types.
- **File System:** Extracted datasets in a neutral format for exchange between systems.
- **Target:** Datasets imported into the DME, which again may have their own data types.

It is often a surprise to people to learn that data can easily be corrupted as it moves between these three stages. Databases, warehouses, and other applications are generally not designed for dumping out their data – they are designed for storing and archiving their data. Administrators rarely encounter a requirement to reverse this process and start moving data out of their systems. The process of writing custom code to turn a dataset into a file with an agreed specification can have bugs. The same applies when the data is moved from a file into the DME. At any of these transitions, data can be corrupted or truncated, thus breaking data provenance.

4.5.2 Guerrilla Analytics Approach

Think of when you get change from a purchase in the shops. The cash register reports what your change should be. The sales assistant takes that cash out of the register and usually counts it in front of you to verify it for them and you. They then hand the cash over to you. You might do another quick count of the notes and coins before you put the cash in your pocket. This is a good analogy for data extraction. The source system (cash register) reports that it has a certain amount of data (cash) for extraction. The system administrator (sales assistant) takes that data out of the system and counts it to verify for them and for you that the expected amount of data was taken out of the system. When you receive the cash, you do another count to make sure you did not drop a coin or perhaps that two notes were not stuck together.

The best way to detect errors in data transfer is to calculate checksums on the data before it leaves its original source system, and again calculate checksums on the data after it is loaded into their target destination in the DME. A checksum is a type of calculation on the data such that if the data changes then the result of the checksum calculation changes. Typical checksums for analytics include sums of values and sums of string lengths. Since checksums are so important to the integrity of data at the very beginning of the Guerrilla Analytics workflow, they are given a more detailed treatment in a later chapter on data testing.

War Story 5: The Moving Goal Post

Mary is a junior data analyst on a project. She was asked to go to a client site and extract data from their accounting system. This was to help the client with their expense audit process. Mary successfully got access to the source system, connected up to the database, and began extracting data out into text files to take back to the analytics lab. She was thinking ahead, so asked to work after office hours when the system was not in use and her work would not be a disruption to the client.

Mary did everything by the book. She meticulously calculated checksums, exported the data, and brought it back to the lab. When Mary began her analyses, she noticed something very strange. The data was not consistent at all. Accounts in one dataset did not exist in another dataset. Transactions appeared in one account and not another.

Mary did not know that while the office workers were not on the expenses system, it was still being used by overnight reconciliation processes. The data was changing as she calculated her checksums and exported the data. Working against such a moving goalpost, there was never any chance that she would have a consistent collection of datasets and so could not produce reliable reports. Somewhat sheepishly, Mary had to go back to the client and ask them to freeze their systems while she reran her data extraction. It was a bit of an own goal for Mary.

4.6 PRACTICE TIP 4: CAPTURE FRONT-END REPORTS

4.6.1 Guerrilla Analytics Environment

Data documentation is often of poor quality or out of date. Even when documentation is available, it is difficult to correctly reverse-engineer an extracted system. How do you know which tables to join together when there are often hundreds? How can you know the business rules that are encoded in the data? Does a coded value like “P” mean purchased or procured?

4.6.2 Guerrilla Analytics Approach

When data is extracted from a system, captured front-end reports are a great help in starting the reverse engineering of a system and validating correct interpretation of the underlying data. There are often many methods and options for generating a report. Think of a report where you generate your bank account transactions in your online banking. You may be able to select a particular month of transactions or a date range. You may also be able to set a transaction size range and a number of accounts. When reports are captured, the application options and login used to generate the front-end report should also be recorded so that the same type of report can be regenerated in the future.

4.6.3 Advantages

A set of front-end reports to accompany a data extraction is useful for two reasons.

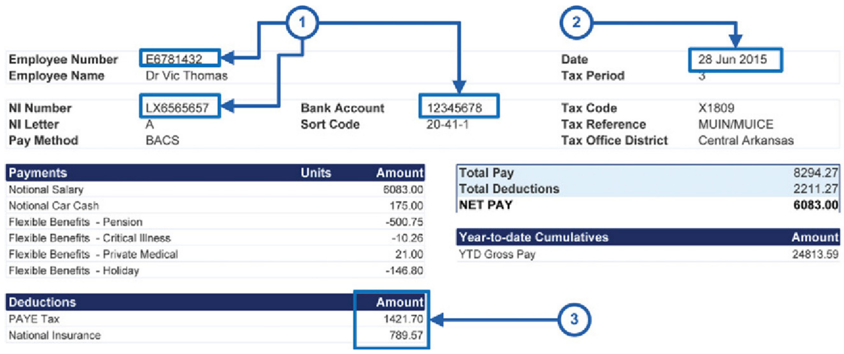
- **Data validation:** Front-end reports are a good check that data has been extracted correctly. Consider a typical ERP system. If the total monthly sales values in the front-end report are the same as the total monthly sales values calculated from the underlying data, then you have high confidence that the data has been extracted correctly and are being interpreted correctly. This is a huge advantage in Guerrilla Analytics projects where documentation is often scarce and unreliable.
- **Reverse engineering:** Front-end reports are for end users and so they present the underlying data in a human friendly format after it has been joined together. Using keywords and values from a report, you have a starting point from which you can begin to explore the underlying data and test its relationships. This is particularly useful when documentation is lacking or is out of date.

Figure 11 shows an illustrative front-end report and some typically useful areas of the report. The example is a pay slip since that is a front-end report we all hopefully see a few times a year. In area 1, there are several unique identifiers for an individual. These are the employee number, National Insurance (NI) number, and the bank account number. These patterns of letters and digits should be unique in the underlying database. This provides you with a starting point in the underlying data from which you can begin to map out relationships to other datasets.

Area 2 is a report date. This could allow you to filter out any data after this date and simplify the scope of the reverse engineering exercise.

Finally, area 3 is an example of self-consistent data. If we can identify all the deductions for an individual in a given time period, then we know what they should add to the total deductions number on the bottom right of the report. Similarly, other totals allow us to find the right subset of "Payments."

Presented with a report like this and underlying data that is consistent with the report at the time of extraction, the analyst can find an entry point into the data. They can then start to build out relationships to payments and deductions using the available lists, totals, and dates on the report.



Employee Number	E6781432			Date	28 Jun 2015
Employee Name	Dr Vic Thomas			Tax Period	3
NI Number	LX6565657	Bank Account	12345678	Tax Code	X1809
NI Letter	A	Sort Code	20-41-1	Tax Reference	MUIN/MUICE
Pay Method	BACS			Tax Office District	Central Arkansas

Payments	Units	Amount
Notional Salary		6083.00
Notional Car Cash		175.00
Flexible Benefits - Pension		-500.75
Flexible Benefits - Critical Illness		-10.26
Flexible Benefits - Private Medical		21.00
Flexible Benefits - Holiday		-146.80

Total Pay	8294.27
Total Deductions	2211.27
NET PAY	**6083.00**

Year-to-date Cumulatives	Amount
YTD Gross Pay	24813.59

Deductions	Amount
PAYE Tax	1421.70
National Insurance	789.57

FIGURE 11 Example front-end report

4.7 PRACTICE TIP 5: SAVE RAW COPIES OF WEB PAGES

4.7.1 Guerrilla Analytics Environment

Not all data is extracted from applications. Increasingly, data is provided on "open data" web platforms such as the United States government's www.data.gov website, social media platforms such as *Twitter*, and on news sites such as *BBC* news. Web scraping is the technique of automatically scanning a web page and extracting its content. This content may be free-form text such as news articles. It may also be structured information as presented in tables. It may be a series of blogs, tweets, comments, and other social data streams. This data can change over time since the providers are not obliged to adhere to your data provenance requirements.

Web scraping is difficult because web pages are usually a fairly unstructured data source. Although tools for scraping are improving, it may still take significant ad hoc program code to extract useful structure from a web page.

What should be done when these data sources feed into the data environment?

4.7.2 Guerrilla Analytics Approach

When scraping web data, make sure to save the original raw web page. Think of these as the raw data files from the "source" system.

4.7.3 Advantages

Following this tip, you have two advantages.

- **Permanent record of volatile data:** You have a permanent record of your raw data (the web page).
- **Revisit and improve:** You can revisit your raw data and reapply scraping processes to it in the event that mistakes are discovered in the scraped data. You can still do this when the original web page no longer exists in the same format or with the same content.

4.8 PRACTICE TIP 6: CONSISTENCY CHECK OCR DATA

4.8.1 Guerrilla Analytics Environment

OCR data is error prone. Letters and numbers can be mixed up by the automated OCR software. This is not a huge problem for text as it is rarely critical that every single word is correct. When numbers are incorrectly OCRed, this can create more serious problems. Misplacing a decimal point can make a difference of millions of dollars to a report.

4.8.2 Guerrilla Analytics Approach

When dealing with OCRed data, look for opportunities to check the data for internal consistency. Internal consistency means that rather than comparing the data to some external source, the data is instead compared to itself to make sure that it is consistent with itself.

For example, if a document has a table of numbers with a total, check that the OCR numbers sum up to the total listed in the report. For text data, run a check against a dictionary to find misspelled and unrecognized words.

4.8.3 Advantage

Running these types of checks gives you more confidence in a low-quality data conversion process and will quickly identify errors in the data OCR process.

4.9 WRAP UP

This chapter discussed the first stage of the Guerrilla Analytics workflow – Data Extraction. You learned about the following topics.

The pitfalls of data extraction are as follows:

- It is impossible to reproduce and checksum extracted data because data was extracted from a moving target.
- Inability to assess completeness of extracted data because checksums were not performed at the right time.
- No reproducibility of data extraction because the program code and front-end report generation options were not recorded.
- There is no reference to compare to because no front-end reports were captured.
- Front-end reports cannot be reproduced because the particular user login and reporting options were not recorded.
- There is no record of raw web pages used in scraping.
- There are no checks of OCR data so that the inevitable errors in the OCR process go undetected.

There are several tips that help avoid these pitfalls and mitigate data extraction risks.

- Put the source system in a frozen state.
- Extract data into an agreed file format.
- Calculate checksums before data extraction.
- Capture front-end reports.
- Save raw copies of scraped web pages.
- Do consistency checks on OCR data.

The next chapter will look at how this extracted data is tracked as it makes its way to the analytics team's environment.