Chapter 18

Technology

18.1 ANALYTICS CAPABILITIES

There are some technologies that are absolutely essential to a successful Guerrilla Analytics team and in addition some secondary technologies that are highly desirable, if time, resources, and circumstances permit. The essential technologies are as follows.

- **Data Manipulation Environment (DME):** This book has referenced a DME throughout. The DME is some place to store and manipulate data. This may be a domain-specific DME such as the R environment (Crawley, 2007) or may be a more generic relational or NoSQL database. The choice between relational and NoSQL will depend on the type of data analyzed. Sadalage and Fowler (2012) give a good overview of NoSQL technologies.
- **Source code control:** Source code control provides the ability to track the version history of program code, roll back to previous versions of work, and highlight changes made to code between versions. Specific software is not strictly necessary for simple version control, and appropriate Guerrilla Analytics approaches to simple version control have been discussed in earlier chapters. That said, source control tools such as Git (Loeliger and McCullough, 2012) make the management of versions of code much easier.
- **A command line:** A command line (sometimes interchangeably called a shell or terminal) is a window where a user can type commands to interact directly with files stored on the underlying operating system. Because the command line is programmable and lightweight, it can be used to explore and interact with large data files on the file system. This is useful for the "quick and dirty" data profiling and manipulation without the overhead of loading all the data into a heavy weight DME. This helps the Guerrilla Analyst under time constraints. Command lines are also useful in the automation and scheduling of other analytics operations, as is typical when a combination of tools and techniques are brought to bear on a problem.
- **A high-level scripting language:** This is a programming language that is abstracted away from the details of the particular computer and operating system. This makes programs quick to develop and easy to understand.

These languages are great for stitching together analytics stages and activities. Typical popular languages include Python (Lutz, 2009), Perl (Christiansen et al., 2000), and PHP (Tatroe and Lerdorf, 2002).

- **Visualization/presentation:** The team needs to have some methods for visualizing data and presenting results back to the customer. Visualization can take a variety of forms from simple charts within spreadsheets through to web applications hosting sophisticated interactive dashboards.
- **A build tool:** A build tool is a tool for automating the configuration of an environment and the execution of program code in that environment, as well as automating various repetitive tasks the team needs to perform. A build tool could automate tasks such as running all analytics build code, archiving data, and creating the setup on the file system and DME for a new work product.
- **Access to the Internet:** If a team is working with data that is online such as social media data, data on web pages, or open datasets, then some form of Internet access is required. Data security concerns often limit this access in analytics projects but there are ways to provide Internet access for the team while mitigating these concerns.
- **Encryption:** Data security is often important in projects. In very fast-paced Guerrilla Analytics projects, good quality encryption software should be available to the team to mitigate risk of data loss during transfer of data to the team and during delivery of results back to the customer.
- **Code libraries for data wrangling:** Guerrilla Analytics involves an incredibly wide range of data manipulation activities on diverse datasets. There are plenty of tools to help with the most common problems. There may be some tools that evolve from within the team because of recurring project needs. Whether you buy your data wrangling functionality or write it in-house, make sure it is easily available to all teams with clearly labeled versions for data provenance.
- **Code libraries for machine learning and statistics:** If your Guerrilla Analytics team intends to do even the most basic descriptive statistics, then some type of code libraries for statistics and machine learning will be required. Nobody should be wasting time writing their own histogram plotting function, and few people are qualified to write their own neural network library. Source these libraries from third parties and make them available to the team.

In addition to these core technologies, life is made a lot easier and even more efficient for a Guerrilla Analytics team, if they have the following secondary technologies at their disposal.

- **A workflow management tool**: There is a lot going on in a busy Guerrilla Analytics project full of disruptions. Workflow management helps track team activities in the Guerrilla Analytics workflow, bugs, and work product reviews.

- **Automated code documentation tools:** Documentation is time-consuming but very necessary. These tools allow the team to automatically generate and publish presentation quality documentation directly from the team code base.
- **Virtual machine capabilities:** Virtual machines allow rapid deployment of standard analytical tools to new projects, prototyping, and testing in a safe sandboxed environment. When an analytics stack is agreed, this can be maintained as a virtual machine image that can be quickly booted up for new projects, saving the team's precious time in standing up their analytics environment.

The following sections now look at each of these technologies in more detail.

18.2 DATA MANIPULATION ENVIRONMENT

As discussed throughout this book, a DME is an absolute necessity for a Guerrilla Analytics team. A DME is simply "some place" where data can be stored, transformed, profiled, and manipulated with program code.

Two typical approaches to a DME environment exist. There is what is referred to as domain-specific tools. These are both dedicated data storage and data manipulation language all packaged together. Examples of this are the R environment (Crawley, 2007) and the Pandas environment (McKinney, 2012). Alternatively, there are generic databases that can be used and enhanced for analytics. Any relational or NoSQL database can serve this purpose. Analysts then interact with the database using some other external programming language or data manipulation languages such as SQL, SPARQL, and XQuery.

18.3 SOURCE CODE CONTROL

At its simplest, source code control is technology that manages and tracks changes to program code, configurations, and documentation. In any Guerrilla Analytics project, the code, documentation, and configuration are changed many times and these changes will be made by more than one analyst. Source code control software keeps track of these changes in multiple versions of program code. It tracks every change that team members make to code, and can roll back to earlier versions of code. More advanced use allows a team member to take an experimental copy of code and try some development "in parallel" to the team before merging their changes back into the team's code. This is critical for the Guerrilla Analyst who is dealing with changing requirements. Specific versions of files can be "tagged" and released as versions to the customer. If there are bugs or questions about older work products delivered to the customer, the version control system lets you roll back and inspect the history of changes to a particular version of a work product.

The availability of version control software makes Guerrilla Analytics so much easier. Work products can be tagged so that older versions can be recovered and rerun. Team members working on the same code base can coordinate better with one another. Individual changes to code can be tracked and traced back to a team member.

18.3.1 Level of Training and Expertise

Source code control is a significant subfield of software engineering. Things can get quite complex on large software projects. A team may have several released and maintained versions of their code base with associated bugs and fixes in each version. Team members will be producing code for interfaces, testing, configuration, as well as core functionality. This is more than is needed in Guerrilla Analytics, so a judgment call must be made on the source code control features that your team will use and the associated training required.

18.4 ACCESS TO THE COMMAND LINE

As mentioned earlier in this chapter, a command line (sometimes interchangeably called a shell or terminal) is a window where a user can type commands to interact directly with files and the underlying operating system. Because command lines are programmable, very lightweight, and efficient, and have been around for a very long time, they typically have a wide range of useful commands available. These commands expose functionality that would be very cumbersome or simply not possible in a graphical windowed environment.

Consider this problem. You receive hundreds of large text files containing phone call data. This data describes a "from" dialing number and a "to" dialed number. These files are scattered in an arbitrary folder tree, so some are one folder deep while others are up to five folders deep. There is no regular pattern to this folder structure. Now, suppose you are asked to quickly summarize all phone numbers in all data files, and determine if any international calls were made. How would you go about this? Here are some of the approaches I've seen analysts undertake.

- Begin trying to open the files in a text editor and search for phone numbers with the editor's "find" command. This is time-consuming, error prone, and many of the files will not open in a text editor because they are too large.
- Begin loading the text files into a DME. This is a bit better since the analyst at least realizes they are dealing with a data problem. But it's overkill. We're really just facing a simple search problem here and do not need the data manipulations that a DME provides.
- Begin writing a script in a language like Python (Lutz, 2009) so you can walk through all the files in the folder tree and use a regular expression to identify phone numbers.

The third approach is much better! This is a Guerrilla Analytics approach because it is quick and simple. But do we really need to fire up a script in a high-level programming language (assuming an interpreter is available for that language in the constrained Guerrilla Analytics environment)?

There is a better solution. The command line has a rich variety of tools available for iterating through files, editing them, finding patterns, and chaining these commands together. Here are some of the things you can do.

- **Edit large text files:** Find and replace arbitrary text in very large files. Example tools for this are *sed* and *awk* (Dougherty and Robbins, 1997).
- **Find patterns:** Find and filter on text patterns using regular expressions. An example tool is *grep* (Bambenek and Klus, 2009).
- **Examine the beginning or the end of a file:** Return a specified number of lines from the start or the end of a file. Example tools are *head* and *tail*.
- **Count the number of words and lines in a file.** An example tool is *wc*.
- **Sort data files.** An example tool is the aptly named *sort*.
- **Find duplicate rows in a file.** The tool *uniq* comes in very useful.
- **Append files:** Join one or more files together end to end. An example tool is *cat.*
- **Strip a column of data out of a file.** An example tool is *cut*.
- **Join files horizontally:** This is the horizontal equivalent of vertically appending files. An example command is *paste*.
- **Download web pages:** The "wget" command allows you to download web pages and follow hyperlinks.
- **Schedule tasks:** "cron" allows you to schedule when some other program runs. This is helpful when some time-consuming or intensive processes need to be set up to run over night.
- **List files and directories:** Commands such as "ls" and "find" allow us to return all files that match names patterns in some or all of a folder tree.
- **Chain together commands.** With "pipes" you can have one command run and then "pipe" its output into the input of another command.

The phone number problem could be solved at the command line with a strategy like the following.

- Iterate through all data files in the folder tree using a directory listing command.
- Search each file for text patterns that look like a phone number using "grep" and redirect these in a single output file that lists the data file name and phone number text using "sed."
- When finished, count the number of *international* phone numbers in the output file using another "grep."

Availability of a command line is simply essential for many Guerrilla Analytics tasks.

18.5 HIGH-LEVEL SCRIPTING LANGUAGE

Command-line scripts are quick and dirty. While they can be fully functional programs in their own right, sometimes you want the extra features and ease of use of a high-level scripting language. This is a programming language that is abstracted away from the details of the computer and operating system. It makes programs quick to develop and easy to understand. These languages are great for stitching together analytics activities. For example, you may want logging, debugging, and the use of third-party libraries for tasks such as parsing XML files and converting PDF files into text.

18.5.1 Which Language to Use?

There are plenty of scripting languages to choose from: Perl (Christiansen et al., 2000), PHP (Tatroe and Lerdorf, 2002), and Python (Lutz, 2009) are some examples. The choice of language should consider the following factors.

- **DME connectivity:** The language should be able to connect to the DMEs your team uses, pull data out of them, or write data into them.
- **Parse file formats:** The language should be able to interact with a range of file formats such as HTML, XML, PDF, email archives, CSV, and office applications. Data often comes in these formats and you want to avail of all the hard work that has been done to usefully expose these data formats with scripting language libraries.
- **Software engineering features:** Many high-level scripts become full-fledged project tools in their own right. It then becomes important that they have associated language features such as logging, testing harnesses, and error trapping for example.

18.6 VISUALIZATION

There is not much point in doing analytics work if you cannot then present your findings. While a simple dataset is sometimes a sufficient medium for communicating a result, in many cases something more sophisticated is required.

At a basic level, a team should have spreadsheet software available. This allows a team to use a spreadsheet's plotting capabilities to visualize data from the DME. Ideally the spreadsheet can connect to the DME to pull datasets out of it.

One of the problems with spreadsheets is that without some significant effort from the analyst, they allow the underlying data to be modified and potentially broken or corrupted. This is a problem for the Guerrilla Analytics who wishes to maintain data provenance. However, some customers want to be able to interact with their data and cut it in various ways to understand it. In these scenarios, it is beneficial to have a dashboarding tool available to rapid prototype visualizations that customers can interact with. Some tools allow dashboards to be packaged up with their data and delivered to a user who does not have access to the DME. This is helpful because it effectively gives your user the visualization flexibility of a spreadsheet, but shields the user from the complexity of the underlying data.

Spreadsheets and dashboards are limited in size to the amount of data that the end user's machine can process. The most flexible and scalable approach to visualization is to host visualizations in web applications. A server does the heavy lifting and preparation of data in response to the user's interactions, and presents the results in lightweight web pages that the user can consume with nothing more than a modern web browser.

18.7 BUILD TOOL

You have seen how Guerrilla Analytics work products frequently require executing several program code files, perhaps in a variety of programming languages. Some of these code files may be command-line scripts. Some may be written in scripting languages. Some may be query files that manipulate data in a DME. In addition to executing code in some defined order, the team will often want to move things around on a file system, perhaps tear down databases, and rebuild them. Perhaps applications need to be deployed on a web server. You probably want to generate your documentation using the tools we described earlier in this chapter to save precious time. There are two problems here. Mundane and repetitive tasks such as moving files around are error prone and do not add direct value. Second, the inevitable variety of tools the team brings to bear on a problem each have their own options, switches, and configurations. Figure 61 shows an example of the command-line options available to the Microsoft SQL Server sqlcmd utility. A Guerrilla Analytics team is too busy to have to remember and retype this information every time they execute some code.

Build tools are used for automating all of these varied types of repetitive and error prone tasks that an analyst needs to do, and abstracting away complex commands that are tiresome to repeatedly type out.

18.8 ACCESS TO THE INTERNET

Access to the Internet may seem like an obvious necessity. However, many analytics environments deny access to the Internet because of perceptions around security of data. Given the current emphasis on unstructured data from social media and the increasing availability of open datasets, it is becoming increasingly important that data analysts can access websites with appropriate web scraping tools. Additionally, most software documentation is hosted online for search by users.

You need to find a setup that allows your team to access the Internet in a secure way with minimal disruption to their normal work flow.

18.9 ENCRYPTION

Data will need to leave the Guerrilla Analytics team, as work products and data will need to be transferred to the team (physically or electronically) when provided by a customer. Losing this data to a third party can have extremely serious consequences for the customer and for the analytics team. To protect data, high-quality encryption software should be available to every team member.

```
sqlcmd
   -a packet_size
   -A (dedicated administrator connection)
   -b (terminate batch job if there is an error)
   -c batch_terminator
   -C (trust the server certificate)
   -d db_name
   -e (echo input)
   -E (use trusted connection)
   -f codepage | i:codepage[,o:codepage] |
o:codepage[,i:codepage]
   -h rows_per_header
   -H workstation_name
   -i input_file
   -I (enable quoted identifiers)
   -k[1 | 2] (remove or replace control characters)
   -K application_intent
   -l login_timeout
   -L[c] (list servers, optional clean output)
   -m error_level
   -M multisubnet_failover
   -N (encrypt connection)
   -o output_file
   -p[1] (print statistics, optional colon format)
   -P password
   -q "cmdline query"
   -Q "cmdline query" (and exit)
   -r[0 | 1] (msgs to stderr)
   -R (use client regional settings)
   -s col_separator
   -S [protocol:]server[\instance_name][,port]
   -t query_timeout
   -u (unicode output file)
   -U login_id
   -v var = "value"
   -V error_severity_level
   -w column_width
   -W (remove trailing spaces)
   -x (disable variable substitution)
   -X[1] (disable commands, startup script,
environment variables and optional exit)
   -y variable_length_type_display_width
   -Y fixed_length_type_display_width
   -z new_password
   -Z new_password (and exit)
```

FIGURE 61 Example arguments for a command-line tool

Agree to an appropriate encryption method and an encryption software, and make this easily available to the team. Remember, a team in a dynamic Guerrilla Analytics project will need to quickly package up work products and deliver them. Encrypting these work products should be one click away so that encryption is not a disruption to work and efficiency.

If data is being transported around on portable media, there is a danger that the media and the data contained on it could get lost. Make sure the team has a process for:

- Encrypting media.
- Wiping data from media as soon as the data has been stored in a secure environment.
- Controlling storage of unused media so that you know which media are in the field and who is responsible for them.

18.10 CODE LIBRARIES FOR DATA WRANGLING

There are many data formats out there. Every web page is different. Much data is still provided in spreadsheets or PDF documents with an endless variety of formatting and layouts. It is important that the team's scripting language and DME are supported by the right libraries to handle these formats.

When evaluating a scripting language and DME for a team, consider these typical functionalities.

- Ability to convert PDF files into structured text.
- Ability to programmatically navigate HTML, XML, JSON, PDFs, office documents, and other common file formats.
- Ability to handle image formats, resizing, and converting them between different formats.
- Availability of other supporting software engineering functionality such as logging services and testing harnesses.
- Overlap with other technology needs such as visualization and machine learning.

18.11 MACHINE LEARNING AND STATISTICS LIBRARIES

Very often, getting the data into shape and doing some descriptive statistics and visualization are enough to complete a Guerrilla Analytics job. However, you may sometimes need to go further and do statistical modeling such as regression, decision trees, and clustering. You may also need to do machine learning such as association rule mining, neural networks, or Bayesian networks. These types of algorithms are difficult to write correctly and to scale. Plenty of others have done this work before anyway.

The team should have some machine learning and statistics libraries available so they can do this more advanced data science. Perhaps this capability is in a dedicated domain language such as the R environment (Crawley, 2007) or you may opt for some combination of libraries in a general programming language such as the Pandas environment has done with the Python language (McKinney, 2012).

18.12 CENTRALIZED AND CONTROLLED FILE SYSTEM

It is essential to have a centralized file server location where all projects are stored. This facilitates archiving older projects and rolling onto new projects. The process of starting up new project folders in this location should be controlled so that the project folder area does not become a chaotic mess.

We saw in an earlier chapter how all data gets stored in one folder location without trying to create complex folder structures. These confuse users and make it difficult to find data. The same principle applies with project folders. Give every project an ID and label, and name its folder with that ID and label. This saves the Guerrilla Analytics team precious time when leveraging existing knowledge and assets from previous projects.

18.13 ADDITIONAL TECHNOLOGY CAPABILITIES

Although the key capabilities might seem extensive, it is quite possible to achieve them with 5–10 pieces of software and access to a command line. Even the most dynamic environments can usually provide these key capabilities. There are some further secondary capabilities that are very helpful but that usually require installation of more niche software. These capabilities are described in this section. If your customer's analytics environment is flexible or you are working from your team's home environment, then get these tools in place.

18.13.1 Workflow Management

Workflow management is about controlling and tracking the team's activities, and the order in which those activities happen. Workflow management was discussed in the chapter on process.

Workflows can quickly become complex, and tracking them for many team members simultaneously is beyond the capability of a spreadsheet. Workflow management tools take care of this. You can specify the types of activities the team engages in, the states those activities can be in, and the transitions between each of those states. While there are no dedicated Guerrilla Analytics workflow trackers, you can choose one of the existing highly customizable software engineering tools and customize it for your team's analytics workflows (Doar, 2011).

18.13.2 Automated Code Documentation

Most work products require some type of code documentation. Think how hard it can be to understand your own code when you return to it after several months. Now imagine what that is like for a customer or another team that is inheriting your work. Guerrilla Analytics focuses on minimizing time wasted in understanding work products during handovers and reviews. Having well-documented code helps with this.

There are two basic aspects to code documentation. The first is some overall documentation that describes any configuration and operation of the code or application, sources of data, business rules, and outputs. The second aspect is documentation of the actual program code files to explain particularly complex pieces of program code, and give an overview of what a code file does.

18.13.2.1 Overall Documentation

This documentation is separate from the technical analytics work, but must be written and maintained in sync with the analytics work. For example, your final report will document data sources, business rules, and key assumptions in your analysis. So while the majority of this documentation is "business level," it must also relate to the latest state and outputs of the technical analytics work. Maintaining this type of documentation is made much easier if the documentation is written in a file format that can be usefully put in source code control alongside the analytics code and outputs. By "usefully" I mean that it is in some type of plain text format that can be compared between versions as opposed to binary document formats.

The obvious solution here is to write documentation in plain text. But plain text is not very pretty and we might be giving documentation to a customer or stakeholder. A great candidate for professional looking documentation that can be maintained in source code control is Markdown (Gruber, 2004). Markdown allows one to create documentation in a format that is easily readable (plain text), and then convert that documentation to a multitude of formats such as HTML and Microsoft Word. This has two advantages. First, the team can maintain documentation in source code control, close to their analytics work with minimal disruption. Second, you can build the documentation into a presentation output format when required, by applying whatever style is most appropriate. Need all headings to be dark blue? Simply change one line of the configuration file. Need all paragraphs to have 1.5-line spacing? Simply change another line of the configuration file.

18.13.2.2 Code Documentation

It is one thing to have overall documentation of the project. This is useful for managers and customers who want to understand the project at a business level. However, the data analysts also need documentation at their level – down in the code files they write and the analytics they produce. When documenting code, some teams make the mistake of keeping separate documentation that sits alongside the code files. This just does not work in practice. It breaks that fundamental Guerrilla Analytics principle of keeping analysts close to their work.

If the team have to step out of code into some separate parallel documentation, you will find that they forget to document or do not bother. Documentation then goes out of date and is effectively useless.

Code should be documented using code comments that sit right beside the program code that the comments describe. This is still a little problematic. It is very difficult to get an overview of the code base so that you can find a particular functionality or know where to get stuck in to understanding the code. This is where automatic code documentation tools such as Javadoc and Doxygen (van Heesch, 1997) excel. These tools read code comments that are tagged in a particular way, extract those comments, and automatically generate documentation in a variety of output formats. As well as producing high-quality code documentation the use of these tools encourages uniformity in code commenting styles – a further time saver for the Guerrilla Analyst.

18.14 WRAP UP

This chapter described the technology needed to enable a Guerrilla Analytics team. The chapter was not prescriptive about particular technologies as these improve and evolve all the time. Instead, this chapter focused on what a Guerrilla Analytics team needs to be able to do with data and the associated technology choices to enable this. After reading this chapter, you should now understand the following.

- **Core capabilities:** There are some essential technological capabilities to make available to the Guerrilla Analytics team. Without these, a team will struggle to be effective in a dynamic project with frequent disruptions. The core capabilities covered the following:
  - **DME:** Provide a DME. You may need to consider domain-specific options or generic databases. You will also need to consider the type of data modeling such as relational and NoSQL.
  - **Source code version control:** The team needs to be able to track revisions to their code, tag versions of their code, and roll back to previous versions of the code. There are many technologies to choose from, but only a subset of functionality is required for the vast majority of data analytics projects.
  - **Access to a command line**: The command line allows powerful automation of data analytics tasks that could not be achieved in a point-and-click windowed environment, and is also useful for quick and dirty analyses without a heavyweight DME. There are a small number of commands the team should be aware of for data analytics. Choose one type of command line that has these minimum capabilities.
  - **High-level scripting language:** Make a scripting language available for quickly writing short programs that are more sophisticated than a command line script and for gluing together various analytics components.

  - **Visualization:** Make sure there is a method for presenting data visually. There are several options that vary in sophistication. Simple charts within spreadsheets are widely available and well-understood. Dashboards are more sophisticated but also more expensive, and require specialist skills. Web applications are the most flexible and scalable but require some complex technology decisions, as well as a significant upskilling of an analytics team.
  - **Encryption:** Make sure the team has software encryption readily available to them and ensure they know how to use it consistently as a team. Put in place a process for controlling external media and wiping its contents as soon as data has been delivered.
  - **Common code libraries:** Establish common code libraries for data wrangling and for machine learning. Make these libraries available in a central location with clearly documented versions.
  - **File system:** Have a central file server where all projects are located and where all data is stored.
- **Additional capabilities:** There are also some additional secondary capabilities that are very beneficial but perhaps a little more difficult to introduce into the usual Guerrilla Analytics environment. The additional capabilities that are good to have are as follows.
  - **Workflow management software:** This helps coordinate what the team is doing, work products that have been delivered, and data that has been received.
  - **Automated code documentation:** This saves time producing reports and descriptions of analyses and code.