Chapter 12

Introduction to Testing

12.1 GUERRILLA ANALYTICS WORKFLOW

Figure 44 shows the now familiar Guerrilla Analytics workflow. In this figure you can see that testing is one of the fundamental building blocks on which all workflow activities are based. Ideally, testing should happen at every stage from Data Extraction all the way through to delivery of work products and reports. The nature of this testing will depend on the stage of the workflow in question. Before exploring this in subsequent chapters, let us clarify what it means to test in an analytics context.

12.2 WHAT IS TESTING?

The main purpose of testing in general is to detect defects so that they may be corrected. In the context of data analytics, defects can include incorrectly programmed data manipulations, incorrectly interpreted and implemented business rules, and incorrect choice of models. Because of the immense variety of conditions in which Guerrilla Analytics work is undertaken, testing cannot establish that everything functions properly under all conditions. Rather you can only establish that it *does not* function under *specific* tested conditions. This is an important, if subtle, distinction. It means that it is very difficult to say that something is 100% correct in all conditions. However, by testing as many combinations of conditions as possible you increase confidence in the correctness of tested work products.

12.3 WHY DO TESTING?

Testing and the resulting discovery of defects have several benefits in Guerrilla Analytics work.

- Increases confidence in the correctness of work products.
- Validates that specific business rules have been implemented as agreed with the customer.
- Validates that visualizations, reports, and other presentations are usable by the customer and give the customer the insights they require.
- Increases confidence in the capability of the team's program code to handle a variety and volume of data.

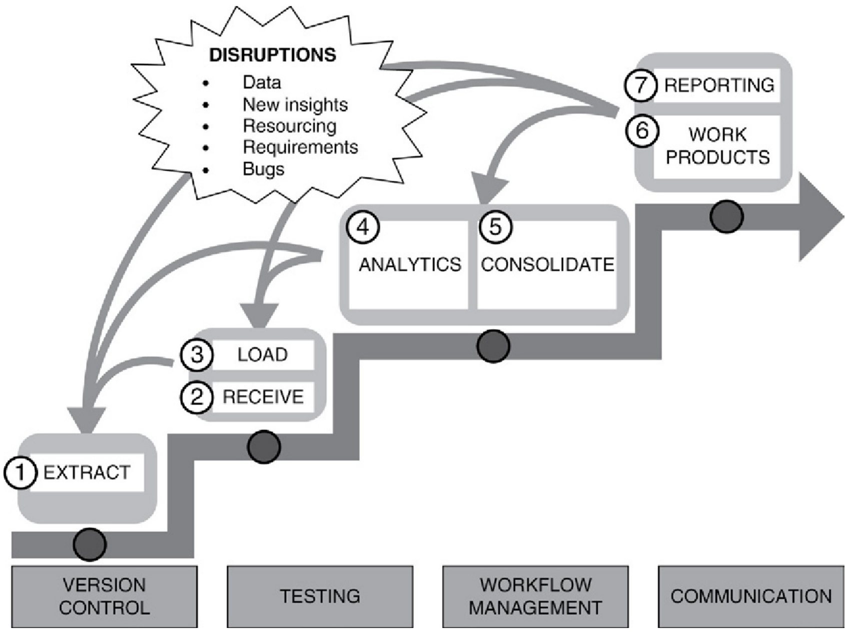


FIGURE 44 **The Guerrilla Analytics workflow**

- Supports the team's development of analytics code by confirming that the code's performance is as designed.
- Increases confidence that a machine learning or statistical model will perform well on new and as yet unseen data.
- Helps identify failures in nonfunctional requirements such as testability, scalability, maintainability, usability, performance, and security.

Testing is well established as a necessary part of traditional software development work. In fact, some software development methodologies even advocate that tests be specified and written before the core product code is written (Beck, 2002). Perhaps because data analytics involves people from a variety of disciplines other than software engineering, there seems to be little awareness or appreciation of the importance of testing in data analytics. Furthermore, the nature of data analytics, being tightly coupled to datasets that are often large, requires a different approach to testing than traditional software development may only involve program code.

12.4 AREAS OF TESTING

There are three overarching areas of testing that need to be done in a Guerrilla Analytics environment. In the order in which they are typically encountered you have:

- **Testing data:** This is testing of raw data to produce a measure of its data quality. Data testing analyzes data to find defects such as corruption of data values, truncation of records, and unexpected ranges of field values. It is all too easy for data to be "broken" as it is extracted from one system and loaded into another system. Data testing increases confidence that this transfer process was executed correctly. Even if data is not officially broken, it may contain features that could trip up subsequent analytics work and so it is useful to detect these features as early as possible.

- **Testing code:** This is the testing that is done to check that analytics program code manipulates data correctly and produces expected data outputs that are sensible and correct. This type of testing is probably closest to software engineering testing as it involves checking the logically correct implementation of analytics program code. However, code testing in Guerrilla Analytics has two major differences to software engineering. First, the data contains flaws that can change with every data refresh and the program code must change to accommodate this. Second, the business rules that are being interpreted and applied in program code are being discovered and evolve with the project.
- **Testing statistical models:** This is the testing that is done to check that a statistical model is well built and a good model of the real world. In some fields, this type of testing is called model validation. There is little point in developing a model if it cannot be used to make decisions about the real world and if the practical and statistical assumptions on which the model is built are not valid.

12.5 COMPARING EXPECTED AND ACTUAL

There are several points to note about the areas of testing described above. Fundamentally, to frame a test you need to do two things.

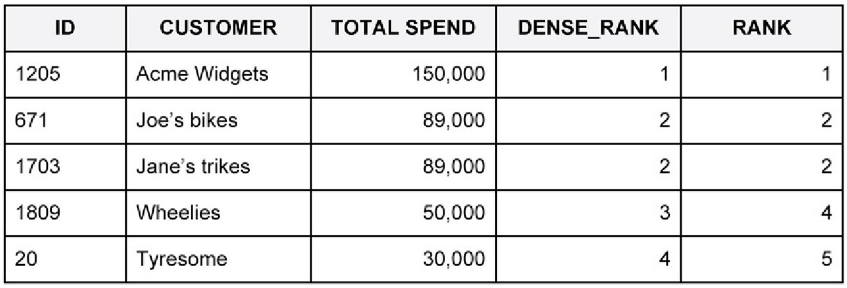
- **Define expectations:** Specify some system's expected output for a given input.
- **Compare:** Compare the system's expected output to its actual output.

Here is what that looks like for some typical examples.

- For some raw data, you might expect it to have 5000 records and no NULL values.
- You might have an expectation from the customer that the data covers some date range.
- You might expect only two values of a particular business domain flag such as customer gender.
- For some simple program code, you might expect it to, say, rank all offices by total spend.

12.5.1 Example

Consider the example data in Figure 45. Customers have been sorted in order of descending total spend. What happens when we apply ranking code to this data? Depending on our choice of ranking function, you may expect several



ID	CUSTOMER	TOTAL SPEND	DENSE_RANK	RANK
1205	Acme Widgets	150,000	1	1
671	Joe's bikes	89,000	2	2
1703	Jane's trikes	89,000	2	2
1809	Wheelies	50,000	3	4
20	Tyresome	30,000	4	5

FIGURE 45 Comparing expected and actual

outcomes. Do you want tied customers to have the same ranking value? What should the next ranking be after these tied customers? The answer depends on the business rule and conversations with the customer. Regardless, you would like to test the code and make sure that it is applying the agreed business rule as expected.

These types of tests become important when you consider what your answer would be if you were asked for the top four customers by spend. Even a simple example like this can raise many questions about expected outputs. This becomes much more complex when dealing with bigger data flows, datasets, and multiple business rules.

12.6 THE CHALLENGE OF TESTING GUERRILLA ANALYTICS

Testing presents many challenges in a Guerrilla Analytics environment.

- **Wide variety of expertise:** Testing of data is familiar to data quality engineers and data governance practitioners. Testing the correctness of program code is familiar to software developers. Testing models is standard practice for statisticians. However, a Guerrilla Analyst needs to be familiar with testing approaches in all of these domains if they are to have confidence in their work.
- **Unknown data:** Since analytics projects usually aim to quantify and understand data in some way, how can we declare expectations about the data? How can you be sure that your estimation of 5% fraud in an insurance claims dataset is realistic and correct? How can you verify that 1000 of your 3 million vendors in a vendor master dataset are duplicates? Increasing confidence in your answers to these types of questions involves some careful thought about data, program code, and business rules.
- **Scale:** The largest software code bases take hours to compile and execute their suites of tests. In data analytics, even a simple sorting of data can take hours to run. Faced with this obstacle of scale and therefore time constraints on code execution time, how do you run sufficient tests to gain confidence in code?

- **Completeness:** In software development, it is possible to clearly specify the input conditions to a program code unit. Techniques such as object-oriented programming (Gamma et al., 1994) help control data and changes in state as they are passed around program code. With data analytics, every record of data is potentially a unique input condition for program code. It is entirely possible for data analytics code to perform as expected on 999,999 records of data in a 1 million record dataset and then fall over on the very last record. How do you design tests that cover this variety of input conditions?

Subsequent chapters will address these challenges. Before diving into these distinct types of testing, there are some considerations that are common across all testing areas.

12.7 PRACTICE TIP 61: ESTABLISH A TESTING CULTURE

People and culture are the first things to get right when testing. Testing is often seen as a "necessary evil" or something that is handed over to a "testing team" to worry about. Testers are then second-class citizens who play catch up with the rock star developers and if bugs sneak through then it's the testers' fault. If such old-fashioned attitudes to testing are allowed to establish themselves then the team will struggle with sentiments such as the following.

- I (a data scientist) cannot test my own code as I know how the code works and am biased. Testing is for a test team.
- We cannot test yet as the data build is not ready. We're too busy adding features and fixing problems. We will test it all at the end.
- I don't have to test my code as it is not customer-facing work so it is ok if there are some small mistakes.
- I don't have to test my code as it is a simple data sample. What could go wrong?!
- I (a developer) am not going to work closely with a test team. Let them read the documentation and figure out what to test.

And so on. You might think this could not happen in your high-performing team but believe me it does. Much of this misunderstanding and unhealthy approach to testing stems from a lack of understanding of testing. You would be surprised how many data analysts have never read anything on testing or heard of testing frameworks. Of course, the lack of good literature specifically targeted at testing data analytics does not help.

The first thing to do is to establish the following in the Guerrilla Analytics team.

- **Everybody who writes code is also a tester.** The burden of quality rests with all those who write analytics code.
- **Testing is about the quality of the team's work products.** Team members need to lose any negative mind-set about testing and being "caught out."

- **Quality is not a distinct function in a team.** Quality is achieved by having development and testing work so closely together as to be indistinguishable from one another.

It is difficult to implement this tip because it is cultural as opposed to being a functional process. The best way to establish an appreciation for testing is through education, feedback, and establishing buy-in from senior analysts who lead and coach others. If you encounter resistance to a test-driven culture it is worth asking whether the individual is so valuable to the team that you can afford to have them waste others' time in understanding and checking their work.

12.8 PRACTICE TIP 62: TEST EARLY

12.8.1 Guerrilla Analytics Environment

There is a temptation to postpone testing under tight timelines and other project pressures and disruptions. "We have important analysis to do, we'll confirm that the data is good once those deadlines are met." This can only lead to trouble. If data is not tested early then the team runs the risk of wasting time producing work products that are not usable because they are based on incorrect data.

12.8.2 Guerrilla Analytics Approach

The Guerrilla Analytics approach is to test early in the development cycle. Of the three testing types, testing data is the most critical and should not be postponed. Testing code can arguably be done after heavily caveated work products have been delivered. Again testing models is critical to the validity of the model. If a model makes poor predictions than it is irresponsible and even dangerous to release it to a customer.

12.8.3 Advantages

The longer testing is postponed, the more expensive uncovered defects become. For example, imagine discovering that a sophisticated statistical model must be scrapped because the data on which it was based was corrupted way back at data import. All the time spent trying out various models and validating them would have been wasted. Testing early helps shape development and reduces future costs of defects.

12.9 PRACTICE TIP 63: TEST OFTEN

12.9.1 Guerrilla Analytics Environment

In the dynamic Guerrilla Analytics environment, data is changing frequently and so the build is changing frequently to incorporate new understanding about this data. Tests can fail to keep pace with build development, meaning they are less useful at guiding development and defects are not detected early enough to be of use.

12.9.2 Guerrilla Analytics Approach

Tests should ideally be run as often as development code changes. A developer should not be changing build code without running the associated tests that cover that code. New features should not be added to the build without sufficient testing. Simply testing around a particular code/data segment is often not enough to know that a build is performing properly. Data flows interact in many complex ways with many dependencies. Ideally, the whole test suite should be run as often as possible on a freshly executed build.

12.9.3 Advantages

Testing often has the following advantages:

- When you identify problems early and often they are much cheaper to fix. This is primarily because fewer work products have been issued that depend on the flawed build.
- Frequent testing discourages the accumulation of "technical debt." That is, testing identifies and highlights defects that need to be fixed for the build to be complete. If these defects are allowed to remain in the code base, they become more difficult to fix over time as the build grows and its interdependencies become more complex.

12.10 PRACTICE TIP 64: GIVE TESTS UNIQUE IDENTIFIERS

12.10.1 Guerrilla Analytics Environment

Tests will incorporate some type of logic and comparisons and so it is difficult to describe them with a short label. Without a better way to refer to tests, it becomes difficult to maintain and understand all tests in the project.

12.10.2 Guerrilla Analytics Approach

Unique identifiers (UIDs) are the best way to identify tests. The UID can be accompanied by as much metadata as is needed by the project. For example, a business description will certainly be helpful. A classification of small, medium, and large is also useful when filtering and searching tests. Tests could also be associated with a particular build layer such as cleaning or business logic.

12.10.3 Advantages

As we saw with data tracking (Section 5.4) and work product tracking (Section 9.5), the advantages of a UID are as follows:

- **Simplicity.** There is a simple and clear way to identify every single test.
- **Tracking.** Every test can be tracked in the file system and in the data manipulation environment (DME).

12.11 PRACTICE TIP 65: ORGANIZE TEST DATA BY TEST UID

12.11.1 Guerrilla Analytics Environment

Test data needs to evolve with test code. This raises the question of how and where test data should be stored and how versions of test data should be recorded. If test data is not managed, it clutters the DME and the file system and distracts the team from the real work of delivery.

12.11.2 Guerrilla Analytics Approach

Test data can be organized in much the same way as ordinary project data. Create a dedicated test data folder for the project. Within this, have a subfolder for each test UID. In each test folder, store clearly marked versions of test data files.

As with ordinary project data, each test file gets loaded into a DME area that can be easily linked back to the test data folder. A good approach in a relational database is a schema named after the test UID, for example.

Figure 46 shows how test data storage could look. The file system on the left contains folders named by Test UID. The DME on the right is almost a mirror

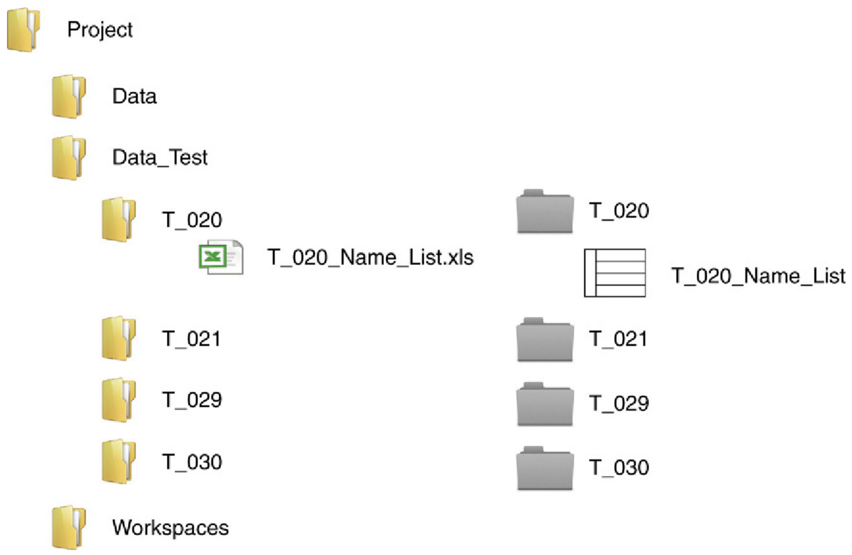


FIGURE 46 Example test data storage

image of the file system. There is a schema for each test UID and a dataset for each test data file. Some tests do not need to call on external data and so are not represented. Some tests may have multiple external datasets and versions of datasets.

12.11.3 Advantages

As you can see from the example of Figure 46, it is very easy to locate data on the file system and DME for a particular test. Clutter is avoided and versions of test data are also accommodated.

12.12 NEXT CHAPTERS ON TESTING

The next chapters in this part of the book will cover the following areas of testing in Guerrilla Analytics.

- **Data testing.** This will cover how to test data that the team receives.
- **Testing builds.** This will cover how to test build code and build datasets.
- **Testing work products.** This will cover how to test the work products produced by the team.

Testing statistical models is a large field in itself that is already well covered in the statistics and machine-learning literature.

12.13 WRAP UP

This chapter was a short introduction to Guerrilla Analytics testing. In this chapter you learned:

- **How testing is fundamental to all aspects of the Guerrilla Analytics workflow.** Testing should be done at data extraction, data import, creation of work products, creation of builds, and model validation.
- **What testing means in the context of data analytics.** Testing is about comparing expected outcomes to actual outcomes to identify defects. Although you cannot cover every possible scenario, increased testing increases your confidence in the correctness of our work.
- **Why testing is done.** At a high level, you want to increase confidence in the correctness of (1) the data the team receives, (2) the program code it uses to manipulate that data, and (3) any statistical models that are applied to the data.
- **How testing analytics presents a variety of challenges.** Expertise is required in a variety of domains. The data is often unknown or poorly understood. The scale of data restricts the amount and coverage of tests that can be executed. The variety of data presents a large number of possible scenarios that must be tested.

- **An appropriate supportive culture must be established to implement testing successfully in a team.** Testing and development must be indistinguishable from one another and destructive attitudes of “them and us” (analysts and testers) must not be permitted.
- **Testing as early as possible in the Guerrilla Analytics workflow saves costs and time.** In the worst case, a statistical model or report that was time-consuming to develop may have to be scrapped because of defects in the data that arose at data extraction and were not identified at any point in the Guerrilla Analytics workflow.