Chapter 15

Testing Work Products

15.1 TYPES OF TESTABLE WORK PRODUCTS

If you think about all the possible work products that a team produces, it may seem like a daunting task to test all of these variations. There are presentations, written reports with embedded analytics, plots, visualization dashboards, data samples, enriched datasets, and many other outputs. However, all of these work products break down into two fundamentally different types of outputs that are tested in two different ways.

- **Ordinary work products:** These are a combination of program code and some output data. This output data could be used in many ways. It may be given to a customer or other team member to work on. It may also feed into a visualization to help the customer with some data insight. It may be one component of a larger report.
- **Statistical models:** These are models in the machine learning or statistical sense and statistical hypothesis tests. That is, they are a relationship between variables that the analytics team has derived or they are a statement about the acceptance or rejection of a hypothesis.

The next sections will describe how these two fundamental types of outputs are tested.

15.2 ORDINARY WORK PRODUCTS

Rather than being overwhelmed by the variety of ordinary work products, think more generally about how these work products are constructed. Some code is written to manipulate data. If a suitable build exists, this code may draw on the convenience datasets that this build provides. If no build exists, then the work product code will probably have to do more of its own data manipulation. Either way, a final set of result datasets is produced. These result datasets may be very small and simple if they are providing a single number for use in a report or some very high-level management information. Alternatively, the result dataset might be a significant data sample that the customer or wider team member wants to use in their own analyses. The result may be a base dataset on which a visualization or plot is constructed. Whatever the output, you have the usual

structure of data manipulation code, result dataset and then some type of presentation layer. It is this code and result dataset that you need to test for defects.

In a manner similar to data testing, we can identify the following "5C" tests of ordinary work products.

- **Completeness:** Was the right subset of data used in the work product?
- **Correctness:** Is the work product code logically correct?
- **Consistency:** Is the work product consistent with previous versions? If not, how did previous versions of this work product differ from the current work product under test? If the work product is based on a build, does the work product data agree with the build data?
- **Coherence:** Does the work product have sufficient IDs to connect it back to its source data? If the work product has several related parts such as a plot and a table of data, are those parts in agreement with one another?
- **aCcountability:** Can you identify the exact versions of all data sources and build datasets used in the work product and can you clearly identify the work product generated by the program code?

The following sections will elaborate on each of these test types.

15.2.1 Testing Completeness

In Chapter 13 on testing data, a lot of time was spent covering the completeness of data delivered to the team. This focused on methods such as checksums to ensure that the data expected from the source was actually the data received and loaded into the Data Manipulation Environment (DME).

Completeness in a work product means that the correct population of data was used for the work product. For example, if you were asked to produce a visualization of sales data for the first quarter, have you clearly filtered by date to the first three months of the year? If you were asked for an analysis of a particular customer subset, have you clearly extracted that specific subset from the available data when producing the work product and were any customers missing?

Testing completeness of a work product therefore involves comparing the population in the work product dataset to the same population in the source raw or build dataset to ensure that the population has not being corrupted or truncated in the course of the work product's code. Sample tests would look like the following.

- **Comparison of the population in the result set to a known expected population.** Consider this example. A customer has asked the team for details on a specific set of registered website users. The specific users were provided by the customer to the team as a list of names. The work product involved sourcing the proper data identifiers for this list of names and then assembling the required data for those users. A work product test did a comparison of the input list from the customer and the names in the final result dataset. It showed that one registered user was missing. A conversation with

the customer could then take place to see whether the provided name was incorrect or to source where the problematic user name came from.

- **Comparison of the population in the result dataset to the same population in the source datasets to check that data fields have not been dropped or corrupted.** This typically occurs when a work product has to enrich a build or raw dataset. In the course of this work, data can be dropped or corrupted. A good work product test will check that this has not happened.

15.2.2 Testing Correctness

Testing correctness of a work product is a much more nebulous endeavor. Unlike when you test data, you do not usually have a specification or business opinion on what the contents and distribution of the data should look like. The work product code is manipulating the data in new ways to reveal new insights, so how do you judge its correctness? For these types of test, you can appeal to all of the excellent work that has been done in traditional software testing.

When testing, you should check to see that data manipulations have been done in a modular way with clearly defined and commented data steps. You should review the program code to check its logic and approach. You should step into some of the intermediate datasets and do completeness tests. You should profile the result dataset to check for any data corruption and do an overall sense check of the results.

This type of testing is often neglected. Analysts tend to blindly trust their code, especially when it runs without errors. Bear in mind that data is full of potential issues and even a single broken field can have a dramatic effect on a data flow.

15.2.3 Testing Consistency

Work products can go through a number of iterations. The underlying build data may change because data is refreshed or because new data features are added to the build over the life of the work product. A customer may request changes to the work product or identify bugs in the data and analysis.

Testing consistency is about identifying and quantifying changes to the work product data that occur between delivered versions of the work product. This type of testing allows you to make statements or answer questions such as the following.

- The population of insurance claims has increased by 567 because our new build of data contained an additional 567 new claims after a data refresh.
- Adding trades from the previous month now increases the total population of trades in the dashboard to 1430, up from 1201.
- The duplicates you reviewed mean we were double counting sales for 20 sales managers and our total reported sales have now decreased by 310,000 USD.

As with consistency testing of data, consistency testing of work products involves comparisons with previous versions of the work product. This is made

all the easier when all versions of work product datasets are stored in the DME or can be reproduced quickly from earlier versions of the work product code. (Section 9.10.)

Consistency tests are straightforward in principle but are often neglected in practice. Be ready to answer the customer's questions on why numbers have changed.

15.2.4 Testing Coherence

When we discussed coherence in data testing, it was all about whether the data could be joined together so that expected relationships in the data model were actually respected. Coherence of work products arises in two areas.

- **Multiple parts:** Very often, a single work product will be delivered in two or more parts. Imagine a data sample of website customer addresses and all their historical addresses. You might present this as one flat table where the website customer name repeats for every one of their addresses. However, your client wants to primarily focus on the latest address and only wants the historical dataset in case questions arise during their analysis. A good approach here is to deliver two datasets. The first dataset is the list of most recent addresses. This is where the client will do their work on customer address review. The second dataset is a historical list of addresses. This dataset is primarily for investigations and look up. If these datasets are coherent, every customer in the latest address dataset should have at least one address in the historical address dataset. The address that appears in the review set should be the latest address in the historical set.
- **Trace to source:** When producing a dataset for use in a report or visualization, it is tempting to include only the data fields that are of interest in the work product and removed supporting fields such as UIDs. This can lead to problems. When questions arise about the work product, there is no easy way to identify and connect back to the source of the work product data. This type of coherence involves including sufficient UID fields in the work product so that the source data can be identified and joined back to if necessary.

An example multiple parts test would do joins between the multiple parts, much like a join test in a data coherence test. This is illustrated in Figure 58. The

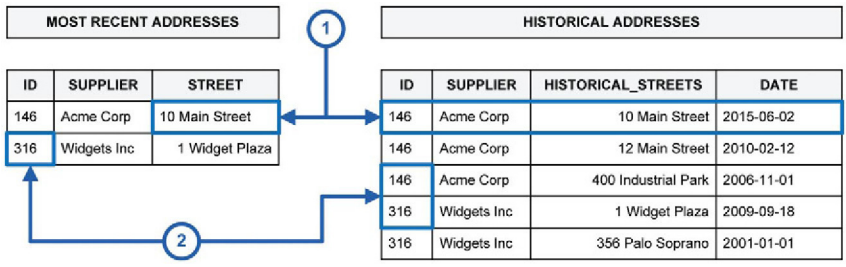

FIGURE 58 Work product coherence test

dataset on the left is the presentation dataset of most recent supplier addresses for two suppliers, Acme Corp and Widgets Inc. On the right is the full historical list of supplier addresses with the as-of date for each address. Acme Corp has had three addresses and Widgets Inc. has had two addresses. There are at least two coherence tests you should do on this data.

1. Is the most recent address for each supplier also the most recent address in the historical dataset?
2. Is every presented supplier in the most recent addresses also in the historical addresses dataset?

15.2.5 Testing aCcountability

The final type of work product test is that of "aCcountability." In work products, aCcountability is essentially about testing data provenance. As was covered previously, the following factors influence data provenance.

- Can all source datasets (from builds or raw data) be easily identified in the work product code?
- Can all result datasets be easily identified and reproduced from work product code?
- Can all outputs (spreadsheets, dashboards, graphics, etc.) be easily identified in the work product's outputs folder?
- Can multiple versions of the work product be easily identified and distinguished?
- Which team member contributed to each of these work products versions?

Testing for work product aCcountability is a matter of reviewing code and inspecting documentation. Ideally you should implement this through training and feedback augmented with the occasional audit of the team's work.

15.3 GENERAL TIPS ON TESTING ORDINARY WORK PRODUCTS

15.3.1 Practice Tip 87: Clearly Identify Work Product Test Code and Data

The team should already be structuring its work product code following Guerrilla Analytics principles. This means the analytics code will be in a folder with a work product UID and will be manipulating data in an area of the DME identified by that same UID.

- **Test code:** Keeping with similar conventions you have seen throughout this book, the test code files should have a similar name to the code files they test so that the files are understood to be related to one another.
- **Test data:** On the DME side, the test datasets reside under the same name space as the work product. This has two advantages. First, the associated test

datasets can be seen right beside the analytics datasets and perhaps delivered along with those analytics results. Second, if the analytics code follows the recommend Guerrilla Analytics practice of tearing down and rebuilding all datasets in its destination location then this will also force a rebuild of the test datasets.

15.3.2 Practice Tip 88: Sense Check Work Product Outputs

It is very difficult to specify test conditions for an ordinary work product. The odds are that this work product has never been created before. Therefore, it is difficult to know what to expect from the analytics code and accordingly what to set as expected in the test. Moreover, many common data mistakes can slip through a work product even if logical tests have passed. It is always advisable to do a quick senses check of outputs. Some common mistakes that arise in a dynamic and pressurized Guerrilla Analytics environment are listed below.

- **Poorly formatted values:** For example, large numbers may be printing in scientific notation or dates may be in datetime format with a redundant time component. This usually happens when exporting data out of DMEs.
- **Truncation at export:** Some analytics environments only print out a limited number of records by default. Does the final output contain all the data it should?
- **Missing data fields:** Are all intended fields included and are unnecessary fields removed?
- **Sensible:** Do the numbers make sense from your knowledge of the business problem?

Code reviews can pick up these mistakes but it is also advisable to train the Guerrilla Analytics team to take a few minutes to quickly sense check their work products.

15.4 TESTING STATISTICAL MODELS

At a high level, statistical models are a relationship between variables in the form of an equation. For example, variables such as customer's age, average current account bank balance, and average salary could be used to model the likelihood that they will default on a credit card balance. The input (or independent variables) of age, balance, and salary are combined as terms in an equation to calculate the probability of defaulting.

The related fields of statistical modeling and machine learning are vast and require trained experts to make judgments about choice of models, choice of variables, and interpretation of model outputs. A detailed discussion of testing is therefore beyond the scope of this book. However, since models and machine learning are often a critical component of analytics work and very much in vogue in "data science," it is worth mentioning some ideas around their testing.

There are two key ways to test a model.

- Model tests.
- Cross validate the model on new data.

The following sections explain these two approaches.

15.4.1 Model Tests

Many models can be tested in specific ways. For example, the quality of a regression model can be tested with statistical tools such as Cook's distance, Q–Q plots, P–P plots, and Box–Cox plots (Montgomery, 2012). Before generating such a model, the independent variables going into the model can be tested for correlations and suitable distributions. Each type of statistical model has its own quality tests and there is no escaping a requirement to be familiar with how to interpret these tests, if not the details of how these tests are derived.

What is equally important is that statistical tests and models are approached with a high degree of skepticism in terms of whether their assumptions are appropriate for the given data. Any model or statistical test which the team publishes should be supported by the necessary tests and data assessments.

15.4.2 Cross Validate Models

Even when statistical quality tests are passed, what ultimately matters is whether a model is a good representation of reality. The canonical way to quality check a model in terms of the quality of its predictions is to perform cross validation.

Cross validation involves partitioning the available data into a training dataset and a test dataset. The training dataset is used to build the model. The test dataset is used to assess how well the model performs on data it has not seen before. K-fold cross validation extends this concept. The available dataset is partitioned into K complementary subsets. The model is trained on K-1 subsets and tested on the remaining subset. This process is repeated for each of the K partitions. The performance on the model against each of the K-test partitions is then combined to give an overall model performance metric.

15.5 GENERAL TIPS ON TESTING MODELS

While model testing is a wide and varied field, there are several general tips that should be followed in a Guerrilla Analytics environment.

15.5.1 Practice Tip 89: Prefer Modeling Tools that Expose Model Data

While it is often helpful to have graphical tools and wizards for model building, these tools can restrict your ability to access the underlying raw data. If

possible, use tools that expose that data. For example, a regression algorithm that allows you to access its residuals data lets you peek under the hood and assess the quality of your model. This gives you much more flexibility and power.

- You can further customize plots and store the output data back in the DME.
- Model outputs can be merged with other data for further analyses and testing.

15.5.2 Practice Tip 90: Store Model Versions for Comparison

When creating a statistical model, it is customary to try several different techniques and variables to determine which is most suited to the given data. For example, when clustering you may try both K-means and Hierarchical-clustering algorithms to see which is best suited to the problem. Although many might treat these early models as "throw away," they are important from the perspective of data provenance. Since the Guerrilla Analytics team must justify its choice of a particular model, the process of evaluating earlier models is important. Store all evaluated models for reference.

15.6 WRAP UP

In this chapter, you have learned about testing work products. In particular, this chapter covered:

- The two main types of work products.
- The five Cs of work product testing.
- The types of testing that can be done on ordinary work products.
- The types of testing that can be done on statistical models.