

TEST-DRIVEN DATA ANALYSIS

Overview of Test-Driven Data Analysis

Test-driven data analysis (TDDA) is an approach to improving the *quality*, *correctness* and *robustness* of analytical processes. It is applicable across the spectrum of data science, from simple reporting and data processing through to the most advanced machine-learning projects, both during development and in production use, whether in batch or as part of a real-time processing workflow.

A Methodology and a Toolset

TDDA is a methodology that can be implemented in many different ways. The TDDA Library and Command-Line tools support data validation, (including constraint generation), automatic test generation for any language, reference testing (semantic tests for analytical pipeline), visual dataset comparison (DataFrame diff), metadata for flat files (tdda.serial, CSVW, and Frictionless. The book covers the entire methodology and includes example data, and code, a comprehensive glossary, test-driven document development (TDDD) and 22 checklists.

Problems Addressed

- problematical input data—poorly specified, missing values, incorrect linkage, outliers, data corruption
- possibility of misapplying methods
- changes to software, libraries or source systems can introduce undetected bugs or changes to results

- changes in distributions of inputs, invalidating previous analytical choices
- reconciliation across multiple or changing data sources.

Reference Tests & Automatic Test Generation

Reproducible research emphasizes the need to capture executable analytical processes and inputs to allow other people to reproduce and verify them. *Reference tests* build on these ideas by also capturing expected outputs and a verification procedure (a “diff” tool) for validating that the outputs are as expected. The `tdda` module extends Python’s `unittest` and `pytest` frameworks to support verification of complex objects (e.g. graphs, files, DataFrames) with exclusions and regeneration of verified reference outputs. Gentest can create Python tests for code in any language.

Constraint Discovery & Verification

There are often things we know should be true of input, output and intermediate datasets, that can be expressed as *constraints*—allowed ranges of values, uniqueness and existence constraints, allowability of nulls, expected structure of textual data (via regular expressions) etc. The Python `tdda` module not only *verifies* data against constraints, but can also *generate* constraints from example data. This significantly reduces the effort needed to capture and maintain constraints as processes are developed and used. Think of constraints as *unit tests for data*.

Utilities

Includes utilities for Dataset Comparison, Unicode, and CSV Data.

How is this misleading data misleading me?

