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. Stochastic Solutions develops an open-source Python module, tdda, which provides tooling support for test-driven data analysis, and incorporates extended TDDA functionality in its commercial Miró analysis software.

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

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 Python tdda module extends the unit test and pytest frameworks to support verification of complex objects (e.g. graphs, files, DataFrames) with exclusions and regeneration of verified reference outputs.

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.

Python TDDA Library

```
pip install tdda
```
or
```
git clone https://github.com/tdda/tdda.git
```

A typical analytical process with the kinds of errors that can occur at each stage. The green percentages are (optimistic) possible success rates for each stage, leading to a 36% success rate first time, falling each time the process is re-used.

Find out more

Watch our TDDA talk from PyCon UK at http://StochasticSolutions.com/tdda-video.
Follow the ongoing developments on our TDDA blog at http://tdda.info and on Twitter at @tdda0.

© Stochastic Solutions Limited 2017–2018 • Version 1.3 • 9 May 2018 • http://www.StochasticSolutions.com • @StochasticSolns