TEST-DRIVEN DATA ANALYSIS

Nicholas J. Radcliffe
Stochastic Solutions Limited
& Department of Mathematics, University of Edinburgh
TEST-DRIVEN DATA ANALYSIS

or

Why should anyone believe your analytical results?

or

Data analysis as if the answers actually mattered
Writing code is not like riding a bike
SOFTWARE DEVELOPMENT (WITH TDD)

Write some (failing) tests*

Write / change code to make tests pass

Simplify code

Add functionality / Fix bugs

Constantly run tests with CI?

Often:
- Well-understood inputs
- Well-understood goal
- Many kinds of errors/failures are unmistakable

* While mocking almost everything
“Don’t mock. It’s not kind.”

— Animal’s People, Indra Sinha.
Why is this lying bastard lying to me?

—Jeremy Paxman
“Shouldn’t we do test-driven data analysis?”

— Patrick Surry, c. 2010
THE BIG IDEA

Transfer the ideas of test-driven development from software development to data analysis (mutatis mutandis*)

*those things which need to be changed having been changed
CORRECTNESS
STURGEON’S REVELATION

“Ninety per cent of everything is crap”

— Theodore Sturgeon, Venture, 1958
We need to extend TDD’s idea of testing for software correctness with the idea of testing for meaningfulness of analysis, correctness and validity of input data, & correctness of interpretation.
DEVELOPMENT PHASE

- **CHOOSE APPROACH**
  - Fail to understand data, problem domain, or methods

- **DEVELOP ANALYTICAL PROCESS**
  - Mistakes during coding

- **RUN ANALYTICAL PROCESS**
  - Use the software incorrectly

- **PRODUCE ANALYTICAL RESULTS**
  - Mismatch between development data or assumptions & deployment data

- **INTERPRET ANALYTICAL RESULTS**
  - Misinterpret results

OPERATIONAL PHASE

- **Using sample/initial datasets & inputs to develop the process**

- **Using the process with other datasets and inputs, possibly having different characteristics**

**ERROR OF INTERPRETATION**
- (bugs)

**ERROR OF IMPLEMENTATION**

**ERROR OF PROCESS**

**ERROR OF APPLICABILITY**

SUCCESS
If you buy into this model, it’s sobering to attach probability estimates to each transition and calculate the probability of success after a few runs . . .
TDDA: LEVEL ZERO

Develop a verification procedure (diff) and periodically rerun: do the same inputs (still) produce the same outputs?

ANALYTICAL PROCESS

INPUTS
DATA & PARAMETERS

Record inputs

ANALYTICAL PROCESS

Capture as scripted, parameterised executable procedure
("reproducible research")

OUTPUTS
DATASETS, NUMBERS, GRAPHS, MODELS, DECISIONS ETC.

Record ("reference") outputs
$ python3 test_graphs.py

File check failed.
Note exclusions: Copyright

FAIL: testTSRGraphs (__main__.TestGraphs)

Traceback (most recent call last):
  File "test_graphs.py", line 111, in testTSRGraphs
    maxPermutationCases=2)
  File "/Users/njr/python/tdda/writabletestCase.py", line 170, in check_file
    self.assertEqual(nFailures, 0)
AssertionError: 1 != 0

Ran 9 tests in 11.320s

FAILED (failures=1)
0 :$ python3 test_graphs.py

........

-----------------------------------------------------------------------
Ran 9 tests in 11.658s

OK
0 :$
TDDA: LEVEL ONE
CONSTRAINTS
## EXAMPLE CONSTRAINTS

<table>
<thead>
<tr>
<th>SINGLE FIELD CONSTRAINTS</th>
<th>DATASET CONSTRAINTS</th>
</tr>
</thead>
<tbody>
<tr>
<td>Age ≤ 150</td>
<td>The dataset must contain field CID</td>
</tr>
<tr>
<td>type(Age) = int</td>
<td>Number of records must be 118</td>
</tr>
<tr>
<td>CID ≠ NULL</td>
<td>One field should be tagged O</td>
</tr>
<tr>
<td>CID unique</td>
<td>Date should be sorted ascending</td>
</tr>
<tr>
<td>len(CardNumber) = 16</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>MULTI-FIELD CONSTRAINTS</th>
</tr>
</thead>
<tbody>
<tr>
<td>Base in {“C”, “G”, “A”, “T”}</td>
</tr>
<tr>
<td>Vote ≠ “Trump”</td>
</tr>
<tr>
<td>StartDate &lt; tomorrow()</td>
</tr>
<tr>
<td>v &lt; 2.97e10</td>
</tr>
<tr>
<td>Height ~ N(1.8, 0.2)</td>
</tr>
<tr>
<td>Z</td>
</tr>
<tr>
<td>---</td>
</tr>
<tr>
<td>1</td>
</tr>
<tr>
<td>2</td>
</tr>
<tr>
<td>3</td>
</tr>
<tr>
<td>4</td>
</tr>
<tr>
<td>5</td>
</tr>
<tr>
<td>6</td>
</tr>
<tr>
<td>7</td>
</tr>
<tr>
<td>8</td>
</tr>
<tr>
<td>9</td>
</tr>
<tr>
<td>10</td>
</tr>
<tr>
<td>11</td>
</tr>
<tr>
<td>12</td>
</tr>
<tr>
<td>13</td>
</tr>
<tr>
<td>14</td>
</tr>
<tr>
<td>15</td>
</tr>
<tr>
<td>16</td>
</tr>
<tr>
<td>17</td>
</tr>
<tr>
<td>18</td>
</tr>
<tr>
<td>19</td>
</tr>
<tr>
<td>20</td>
</tr>
</tbody>
</table>

declare (field-exists "Z")
declare (field-exists "Name")
declare (field-exists "Symbol")
declare (field-exists "Period")
declare (field-exists "Group")
declare (field-exists "ChemicalSeries")
declare (field-exists "AtomicWeight")
declare (field-exists "Etymology")
declare (field-exists "RelativeAtomicMass")
declare (field-exists "MeltingPointC")
declare (field-exists "MeltingPointKelvin")
declare (field-exists "BoilingPointC")
declare (field-exists "BoilingPointF")
declare (field-exists "Density")
declare (field-exists "Description")
declare (field-exists "Colour")
declare (= (length (field-names)) 16)
if (field-exists "Z")
    declare (= (type Z) "int")
    declare (>= (min Z) 1)
    declare (<= (max Z) 118)
    declare (= (countnull Z) 0)
    declare (non-nulls-unique Z)
fi

if (field-exists "Name")
    declare (= (type Name) "string")
    declare (>= (min (length Name)) 3)
    declare (<= (min (length Name)) 13)
    declare (= (countnull Name) 0)
    declare (non-nulls-unique Name)
fi

if (field-exists "Symbol")
    declare (= (type Symbol) "string")
    declare (>= (min (length Symbol)) 1)
    declare (<= (min (length Symbol)) 3)
    declare (= (countnull Symbol) 0)
    declare (non-nulls-unique Symbol)
fi
if (field-exists "Period")
  declare (= (type Period) "int")
  declare (>= (min Period) 1)
  declare (<= (max Period) 7)
  declare (= (countnull Period) 0)
fi

if (field-exists "Group")
  declare (= (type Group) "int")
  declare (>= (min Group) 1)
  declare (<= (max Group) 18)
fi

if (field-exists "ChemicalSeries")
  declare (= (type ChemicalSeries) "string")
  declare (>= (min (length ChemicalSeries)) 7)
  declare (<= (min (length ChemicalSeries)) 20)
  declare (= (countnull ChemicalSeries) 0)
  declare (= (countzero (or (isnull ChemicalSeries) (in ChemicalSeries (list "Actinoid" "Alkali metal" "Alkaline earth metal" "Halogen" "Lanthanoid" "Metalloid" "Noble gas" "Nonmetal" "Poor metal" "Transition metal")))))) 0)
fi
<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Min</th>
<th>Max</th>
<th>Nulls</th>
<th>Sign</th>
<th>Unique</th>
<th>Values</th>
</tr>
</thead>
<tbody>
<tr>
<td>Z</td>
<td>int</td>
<td>1</td>
<td>118</td>
<td>no nulls</td>
<td>pos</td>
<td>yes: 118 / 118 unique (100.00%)</td>
<td></td>
</tr>
<tr>
<td>Name</td>
<td>string</td>
<td>length 3</td>
<td>length 13</td>
<td>no nulls</td>
<td></td>
<td>yes: 118 / 118 unique (100.00%)</td>
<td></td>
</tr>
<tr>
<td>Symbol</td>
<td>string</td>
<td>length 1</td>
<td>length 3</td>
<td>no nulls</td>
<td></td>
<td>yes: 118 / 118 unique (100.00%)</td>
<td></td>
</tr>
<tr>
<td>Period</td>
<td>int</td>
<td>1</td>
<td>7</td>
<td>no nulls</td>
<td>pos</td>
<td>no: 7 / 118 unique (5.93%)</td>
<td></td>
</tr>
<tr>
<td>Group</td>
<td>int</td>
<td>1</td>
<td>18</td>
<td>28 nulls(23.73%)</td>
<td>pos</td>
<td>no: 18 / 90 unique (20.00%)</td>
<td></td>
</tr>
<tr>
<td>ChemicalSeries</td>
<td>string</td>
<td>length 7</td>
<td>length 20</td>
<td>no nulls</td>
<td></td>
<td>no: 10 / 118 unique (8.47%)</td>
<td></td>
</tr>
<tr>
<td>AtomicWeight</td>
<td>real</td>
<td>1.007946</td>
<td>294.0</td>
<td>1 null (0.85%)</td>
<td>pos</td>
<td>&quot;Actinoid&quot; &quot;Alkali metal&quot; &quot;Alkaline earth metal&quot; &quot;Halogen&quot; &quot;Lanthanoid&quot; &quot;Metalloid&quot; &quot;Noble gas&quot; &quot;Nonmetal&quot; &quot;Poor metal&quot; &quot;Transition metal&quot;</td>
<td></td>
</tr>
<tr>
<td>Etymology</td>
<td>string</td>
<td>length 4</td>
<td>length 53</td>
<td>1 null (0.85%)</td>
<td></td>
<td>no: 114 / 117 unique (97.44%)</td>
<td></td>
</tr>
<tr>
<td>RelativeAtomicMass</td>
<td>real</td>
<td>1.007946</td>
<td>294.0</td>
<td>1 null (0.85%)</td>
<td>pos</td>
<td></td>
<td></td>
</tr>
<tr>
<td>MeltingPointC</td>
<td>real</td>
<td>-258.975000</td>
<td>3675.0</td>
<td>20 nulls (16.95%)</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>MeltingPointKelvin</td>
<td>real</td>
<td>14.200000</td>
<td>3948.0</td>
<td>20 nulls (16.95%)</td>
<td>pos</td>
<td></td>
<td></td>
</tr>
<tr>
<td>BoilingPointC</td>
<td>real</td>
<td>-268.930000</td>
<td>5596.0</td>
<td>20 nulls (16.95%)</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>BoilingPointF</td>
<td>real</td>
<td>-452.070000</td>
<td>10105.0</td>
<td>20 nulls (16.95%)</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Density</td>
<td>real</td>
<td>0.000089</td>
<td>41.0</td>
<td>5 nulls (4.24%)</td>
<td>pos</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Description</td>
<td>string</td>
<td>length 1</td>
<td>length 83</td>
<td>66 nulls (55.93%)</td>
<td></td>
<td>no: 36 / 52 unique (69.23%)</td>
<td></td>
</tr>
<tr>
<td>Colour</td>
<td>string</td>
<td>length 4</td>
<td>length 80</td>
<td>85 nulls (72.03%)</td>
<td></td>
<td>no: 30 / 33 unique (90.91%)</td>
<td></td>
</tr>
</tbody>
</table>
Gregory (Scotland Yard detective): “Is there any other point to which you would wish to draw my attention?”
Holmes: “To the curious incident of the dog in the night-time.”
Gregory: “The dog did nothing in the night-time.”
Holmes: “That was the curious incident.”

— Silver Blaze, in Memoirs of Sherlock Holmes
Arthur Conan Doyle, 1892.
Correct interpretation: Zero
Error of interpretation: Letter “Oh”