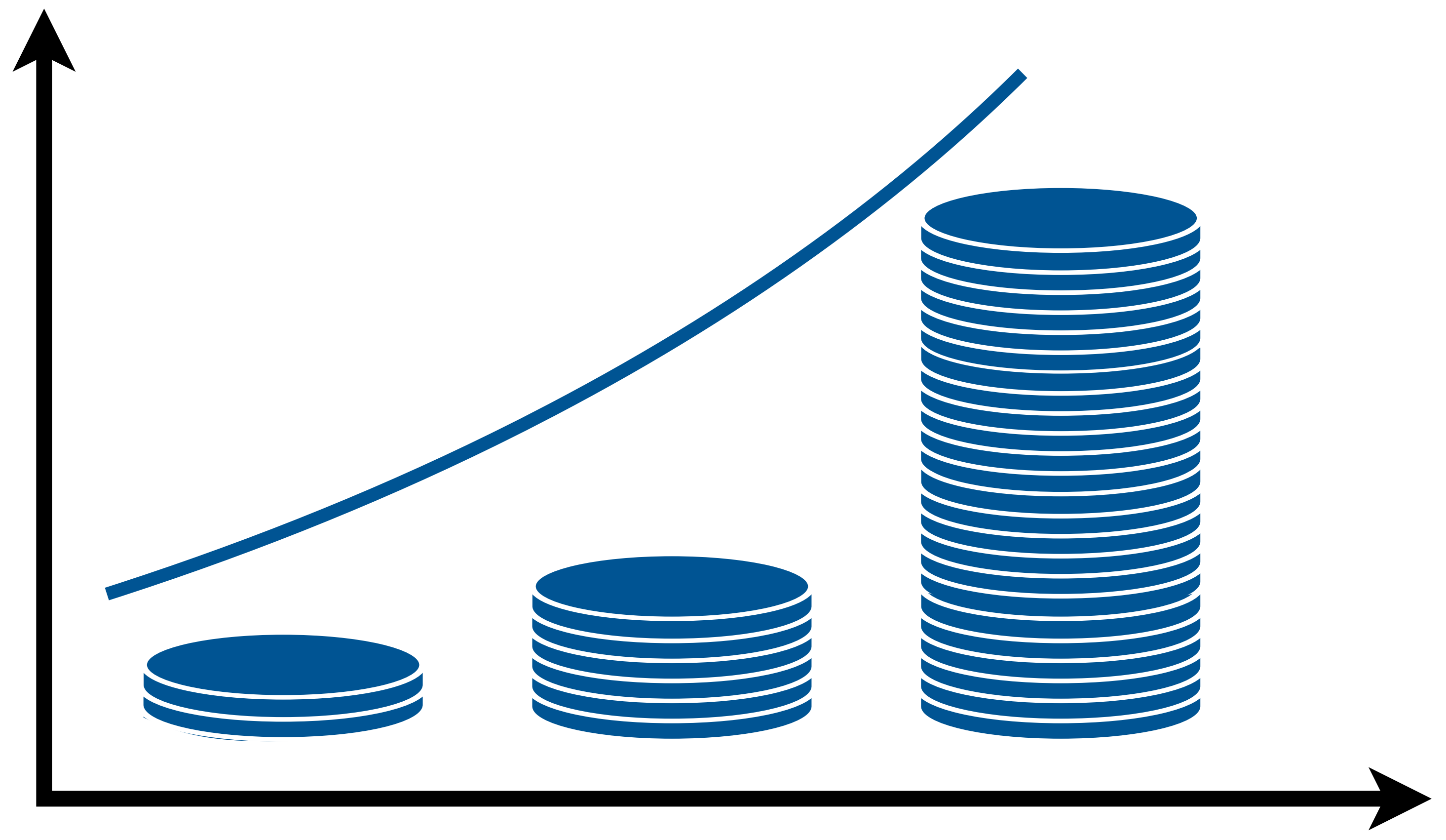
THE SCIENCE OF BAD DATA

DataTech 2019 • DataFest • 14th March 2019

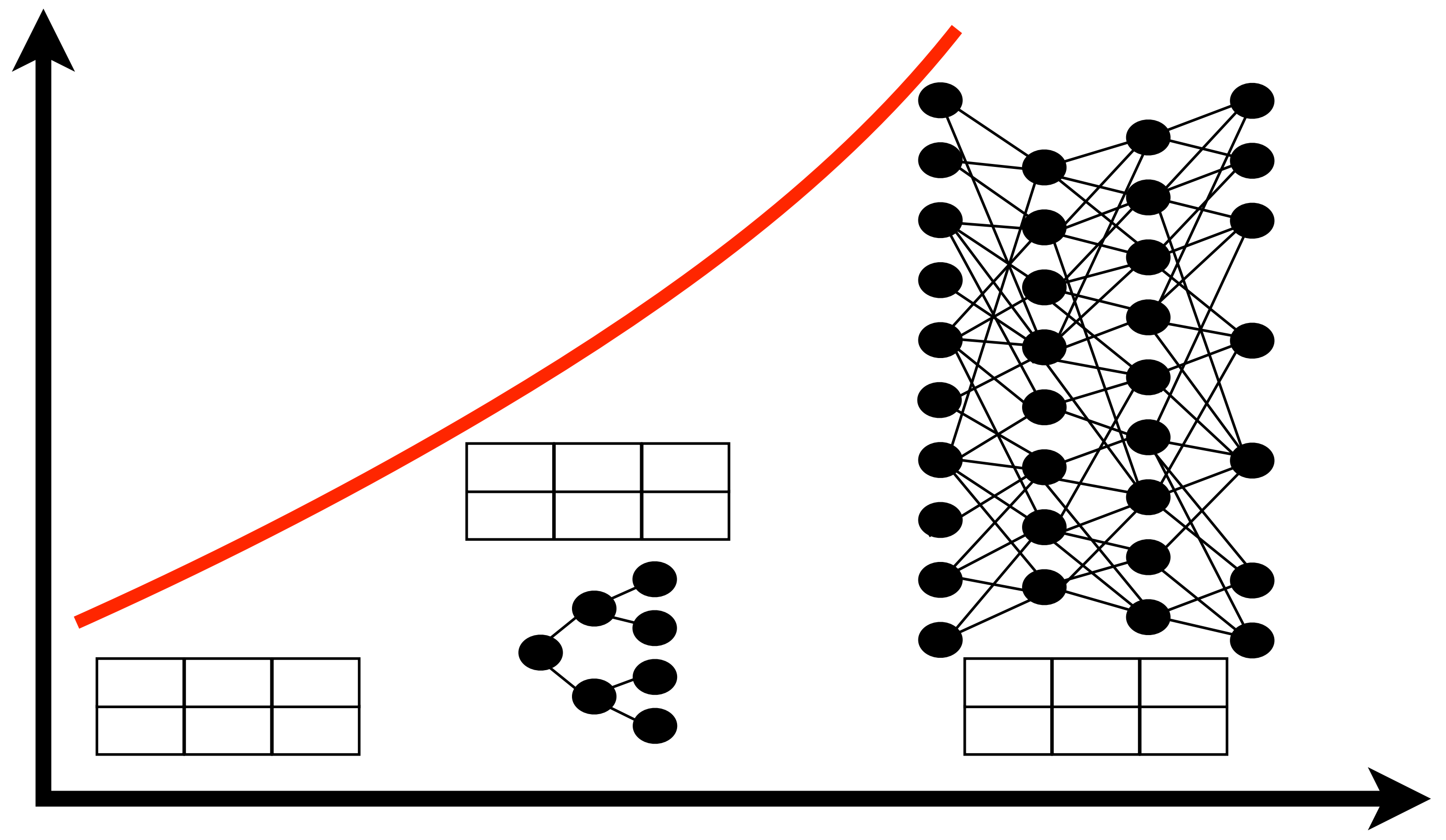
http://stochasticsolutions.com/pdf/science-of-bad-data-datatech-2019.pdf

Nicholas J. Radcliffe
Stochastic Solutions Limited
& Department of Mathematics, University of Edinburgh

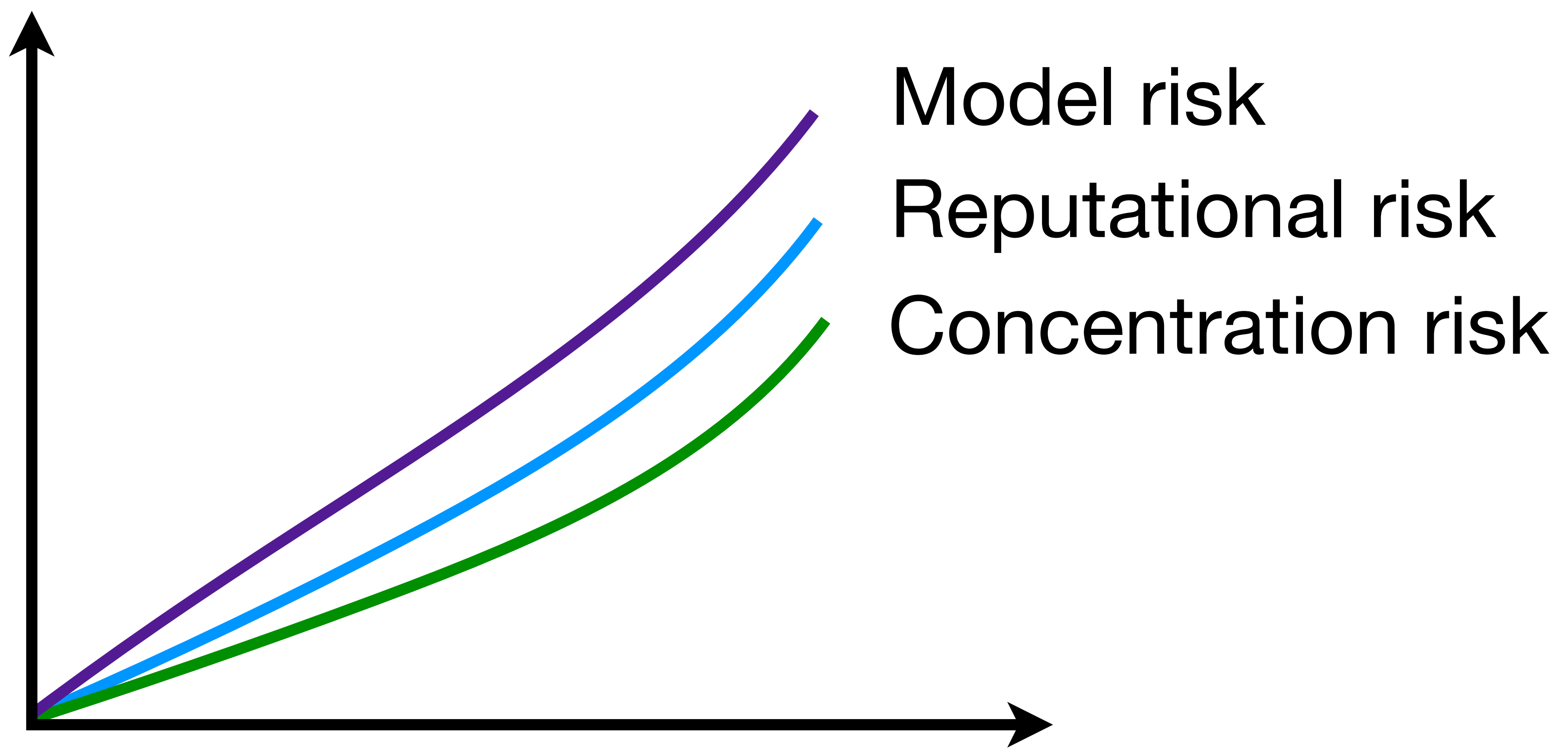
AUTOMATION RISKS



DATA VOLUME

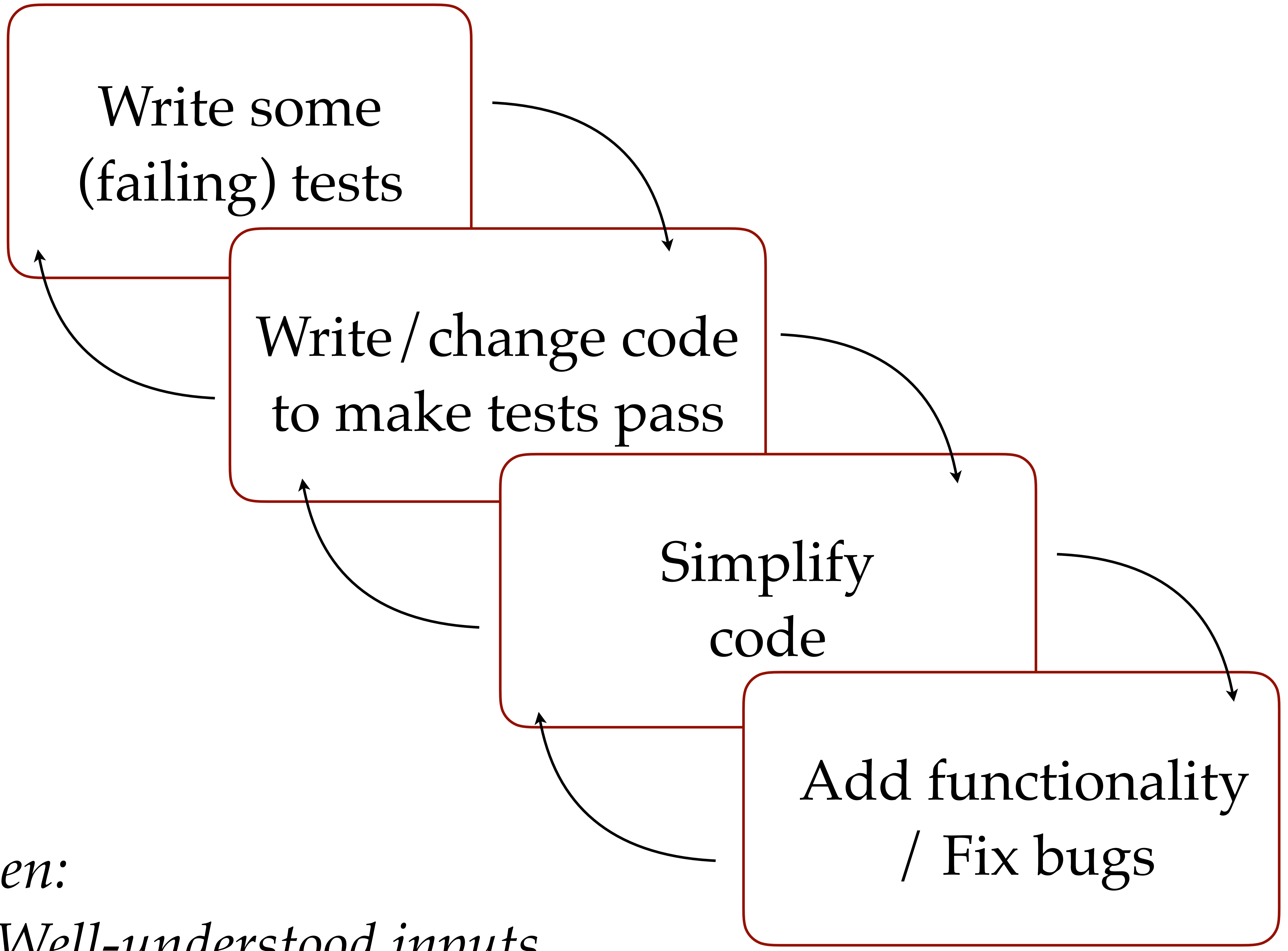


AUTOMATED DECISIONS



SOFTWARE DEVELOPMENT (WITH TDD*)

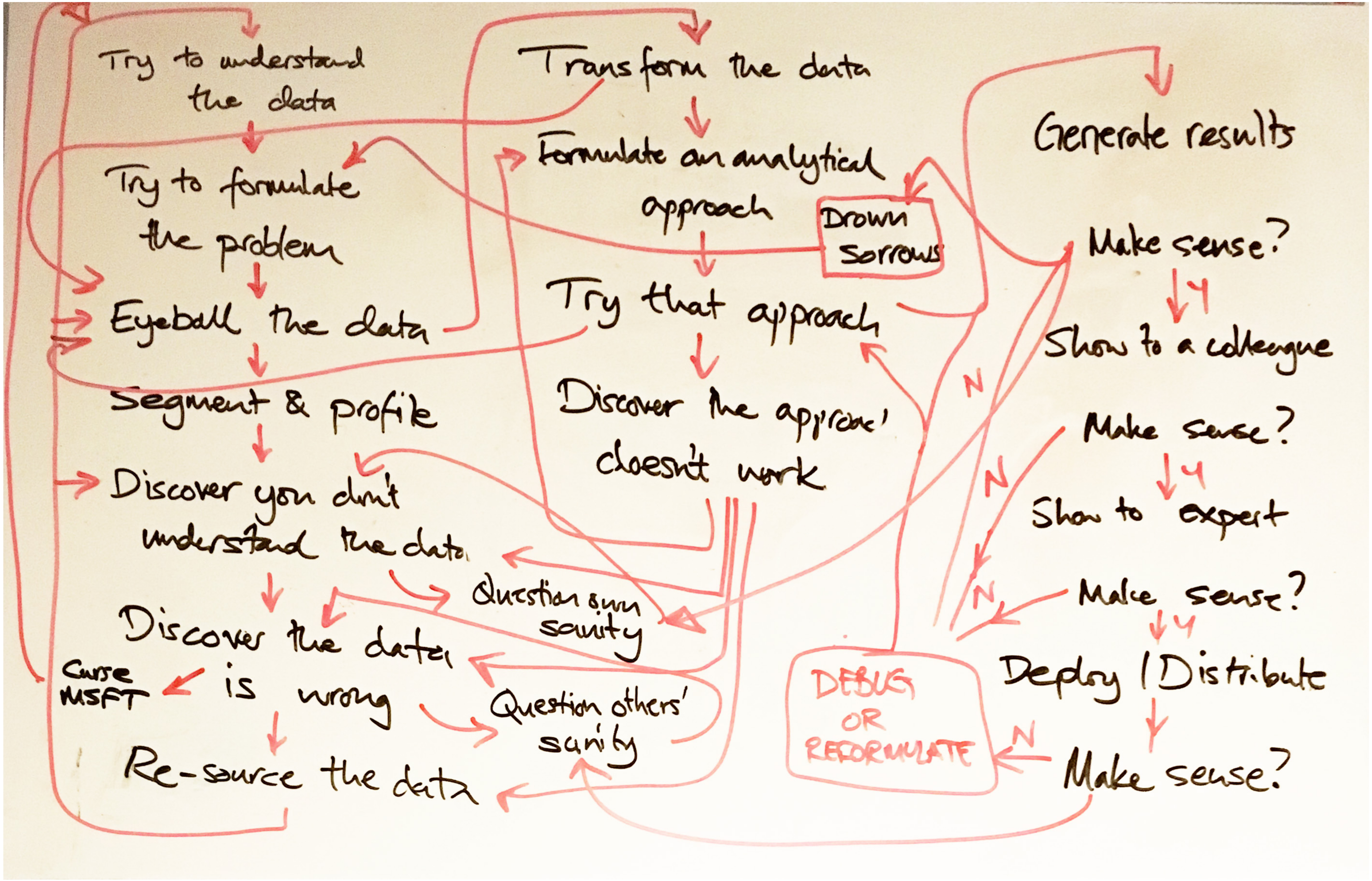
**test-driven development*


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*


Try to understand the data
Try to formulate the problem
Eyeball the data
Segment & profile
Discover you don't understand the data
Discover the data is wrong
Curse MSFT
Re-source the data
Question own sanity
Question others' sanity
Transform the data
Formulate an analytical approach
Drown sorrows
Try that approach
Discover the approach doesn't work
DEBUG OR REFORMULATE
Generate results
Make sense?
Y
Show to a colleague
N
Make sense?
Y
Show to expert
N
Make sense?
Y
N
Deploy / Distribute
Make sense?
N

Why is this
lying bastard
lying to me?

— Jeremy Paxman

TDD $\mapsto$ TDDA

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 and output data,

& correctness of interpretation.

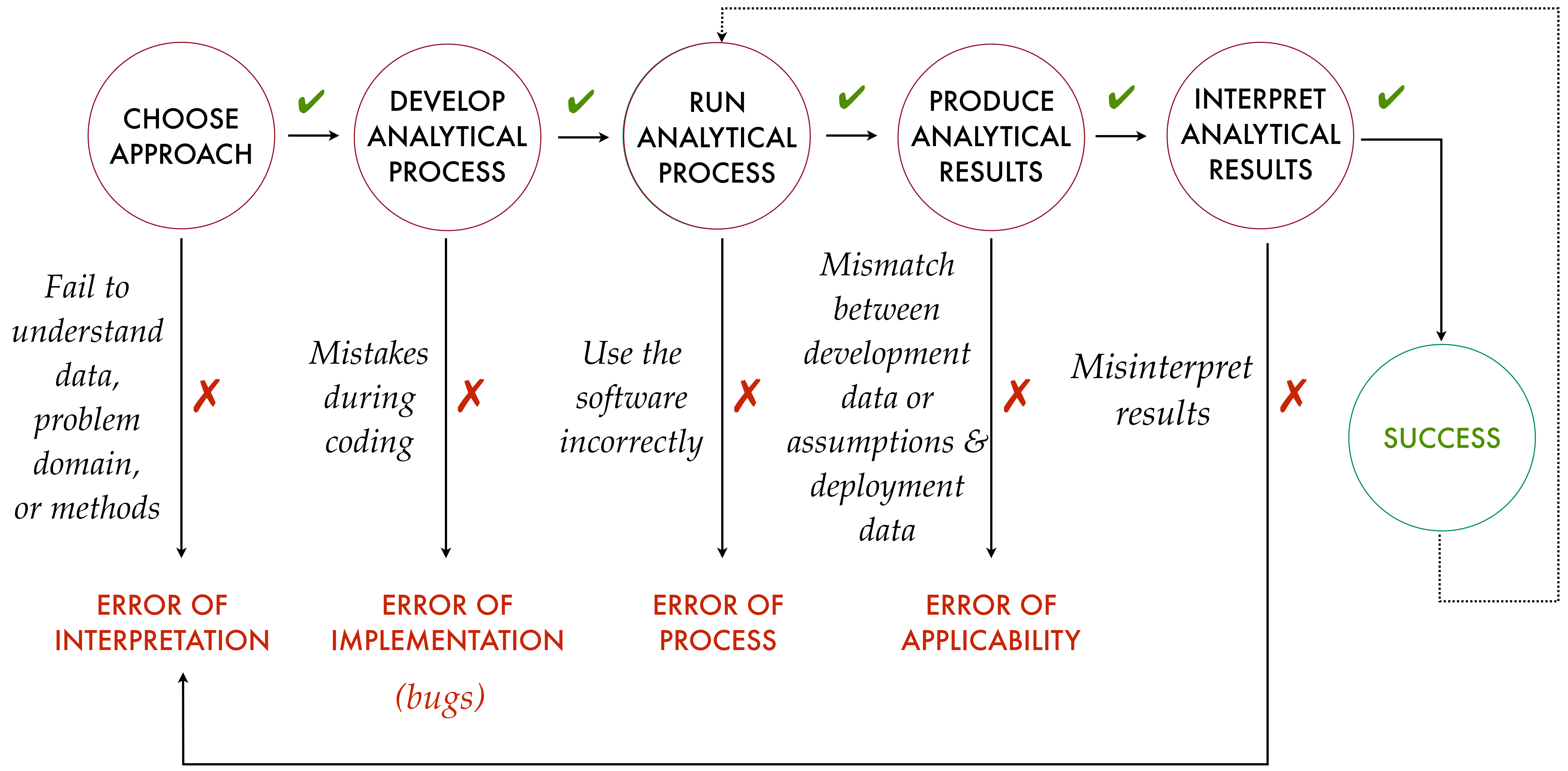
"test-driven data analysis"

DEVELOPMENT PHASE

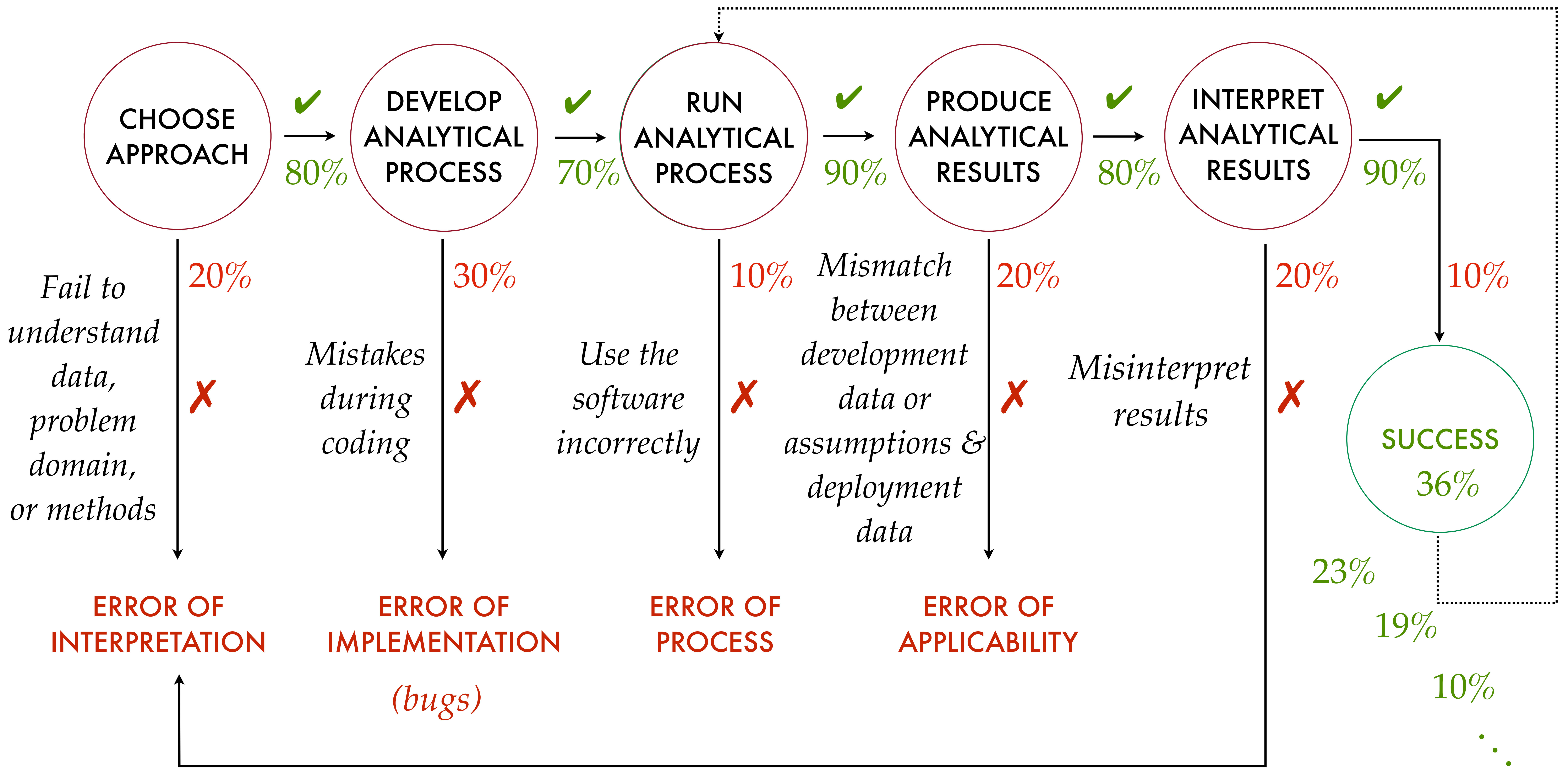
Using sample/initial datasets & inputs to develop the process

OPERATIONAL PHASE

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


CHOOSE APPROACH
✔
DEVELOP ANALYTICAL PROCESS
✔
RUN ANALYTICAL PROCESS
✔
PRODUCE ANALYTICAL RESULTS
✔
INTERPRET ANALYTICAL RESULTS
✔
SUCCESS
Fail to understand data, problem domain, or methods
✗
Mistakes during coding
✗
Use the software incorrectly
✗
Mismatch between development data or assumptions & deployment data
✗
Misinterpret results
✗
ERROR OF INTERPRETATION
ERROR OF IMPLEMENTATION
(bugs)
ERROR OF PROCESS
ERROR OF APPLICABILITY

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 . . .



Error

Garbage In ● ~~*Garbage*~~ *Out*

Warning

Garbage In • Gospel Out

Gold In • Garbage Out

TEST-DRIVEN DATA ANALYSIS: MAIN IDEAS

1. Constraint Discovery & Verification
2. Reference Tests

    2a. Automatic Test Generation (currently in alpha)

TDDA: MAIN IDEAS

1. Constraint Discovery & Verification
    - a bit like unit tests for data
    - can cover inputs, outputs and intermediate results
    - automatically discovered
    - Use as part of analysis to verify inputs, outputs and intermediates (as appropriate)
2. "Reference" Tests
    - *cf*. system/integration tests in TDD
    - With support for exclusions, regeneration, helpful reporting etc.
    - Re-run these tests *all the time*, *everywhere*

    2a. Automatic Test Generation (currently in alpha)
    - Give **`tdda gentest`** a command/script to run.
    - It generates tests for you.

TDDA LIBRARY

Install from PyPI (recommended)

```
pip install tdda
```

or from Github (source)

```
git clone https://github.com/tdda/tdda.git
python setup.py install
```

TDDA LIBRARY

- Runs on Python 2 & Python 3, Mac, Linux & Windows, under `unittest` and `pytest`
- MIT Licensed
- Documentation:
  - Sphinx source in `doc` subdirectory
  - Built copy at `http://tdda.readthedocs.io`
- Quick reference:

  `http://www.tdda.info/pdf/tdda-quickref.pdf`

CONSTRAINT GENERATION, VERIFICATION & ANOMALY DETECTION

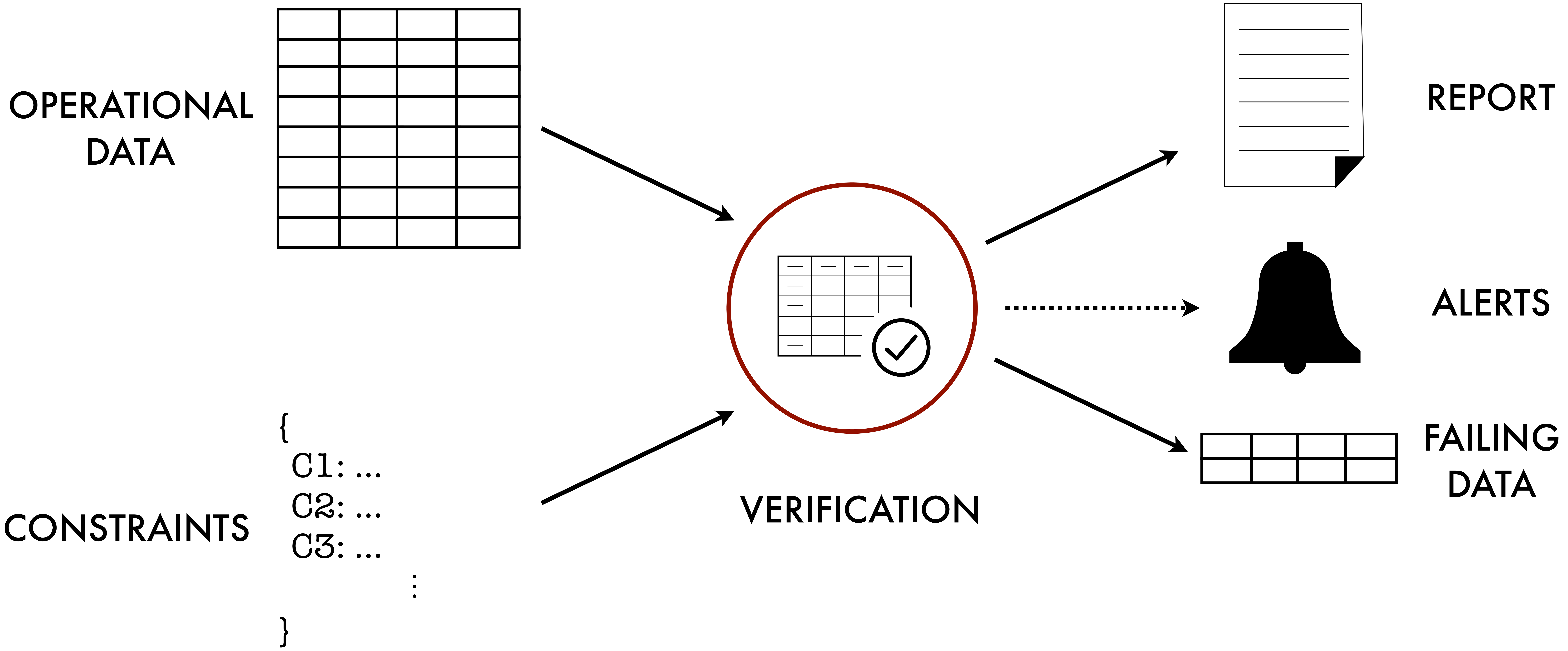
CONSTRAINTS

- Very commonly, data analysis uses data tables (e.g. DataFrames, RDBMS tables) as inputs, outputs and intermediate results
- There are many things we know (or at least expect) to be true about these data tables
- *Could* write down all these expectations as constraints and check that they are actually satisfied during analysis . . . *but life's too short!* (Also: humans are rather error-prone)

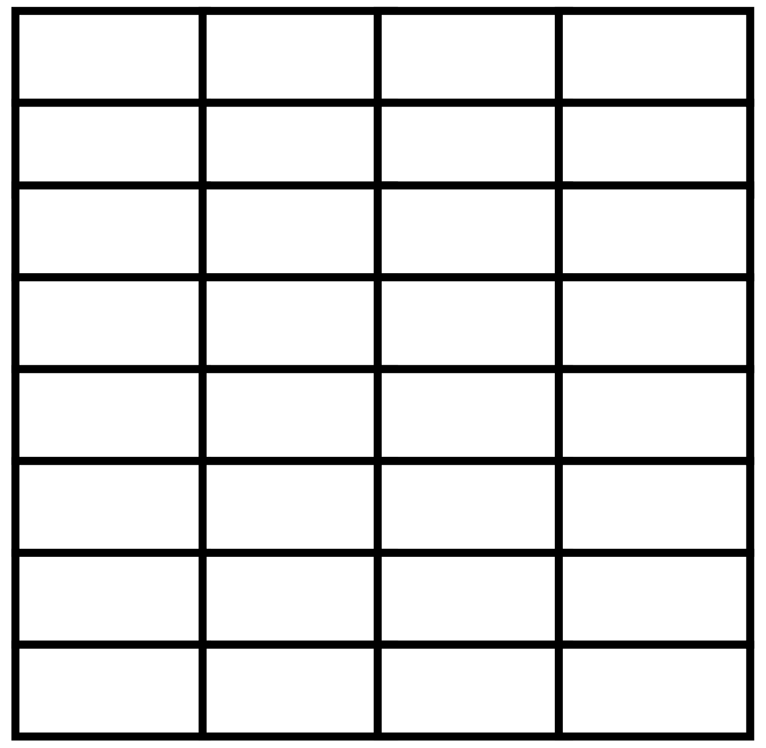
THE BIG IDEA

- Get the computer to discover constraints satisfied by example datasets automatically.
- Verify against these constraints, modifying as required
- (Humans much happier to make tweaks than start from scratch)

DATA VERIFICATION

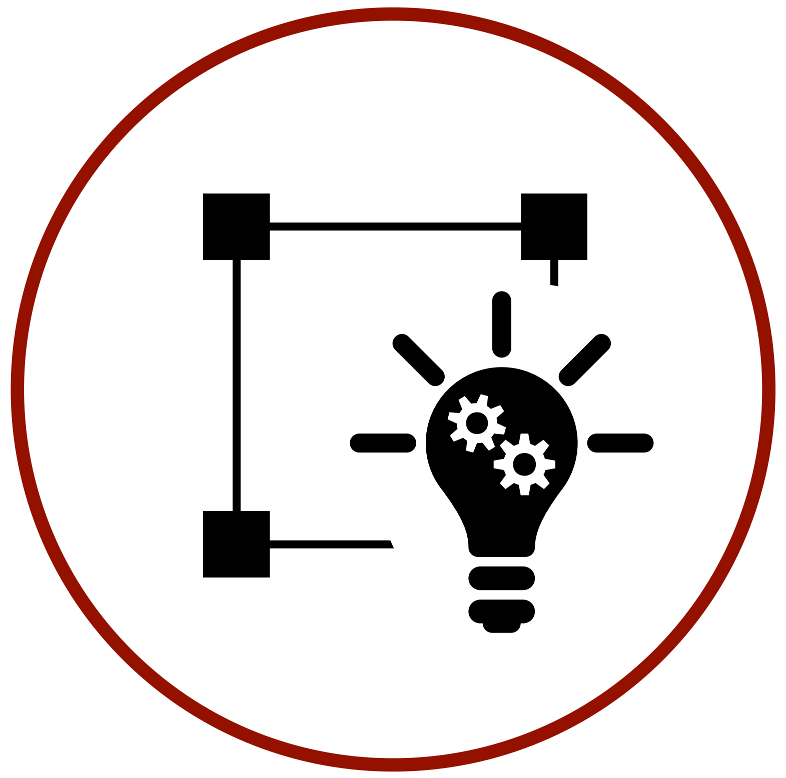


AUTOMATIC CONSTRAINT GENERATION



TRAINING DATA

(believed to be "good")

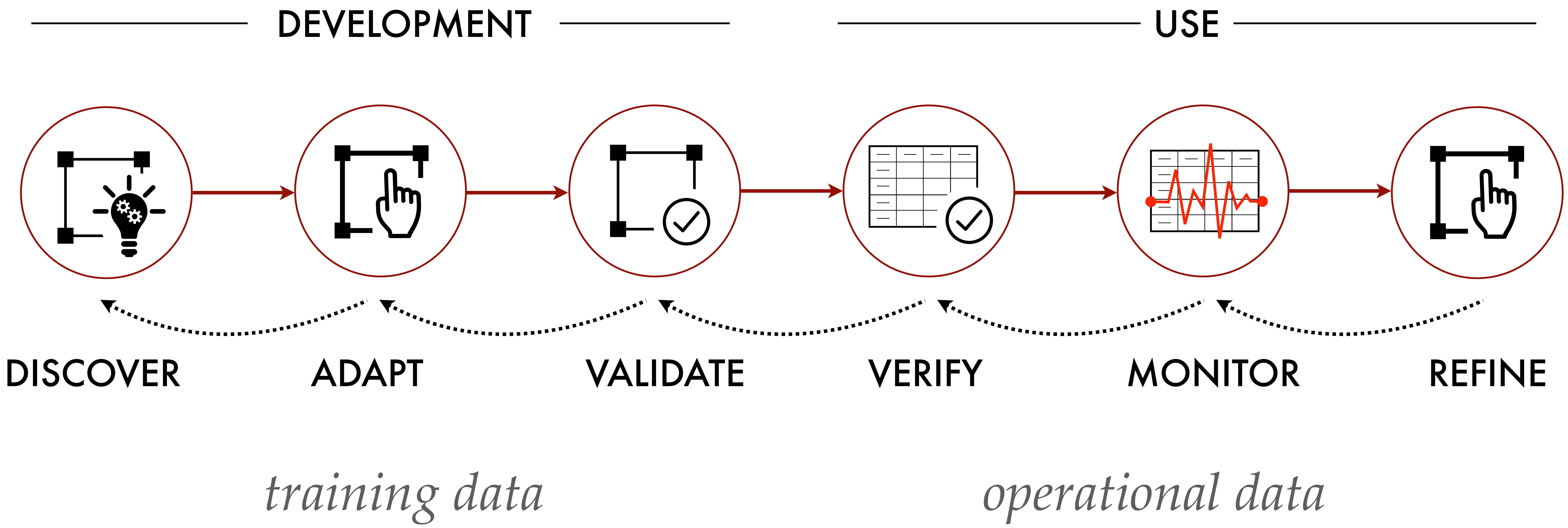


AUTOMATIC DISCOVERY OF CONSTRAINTS

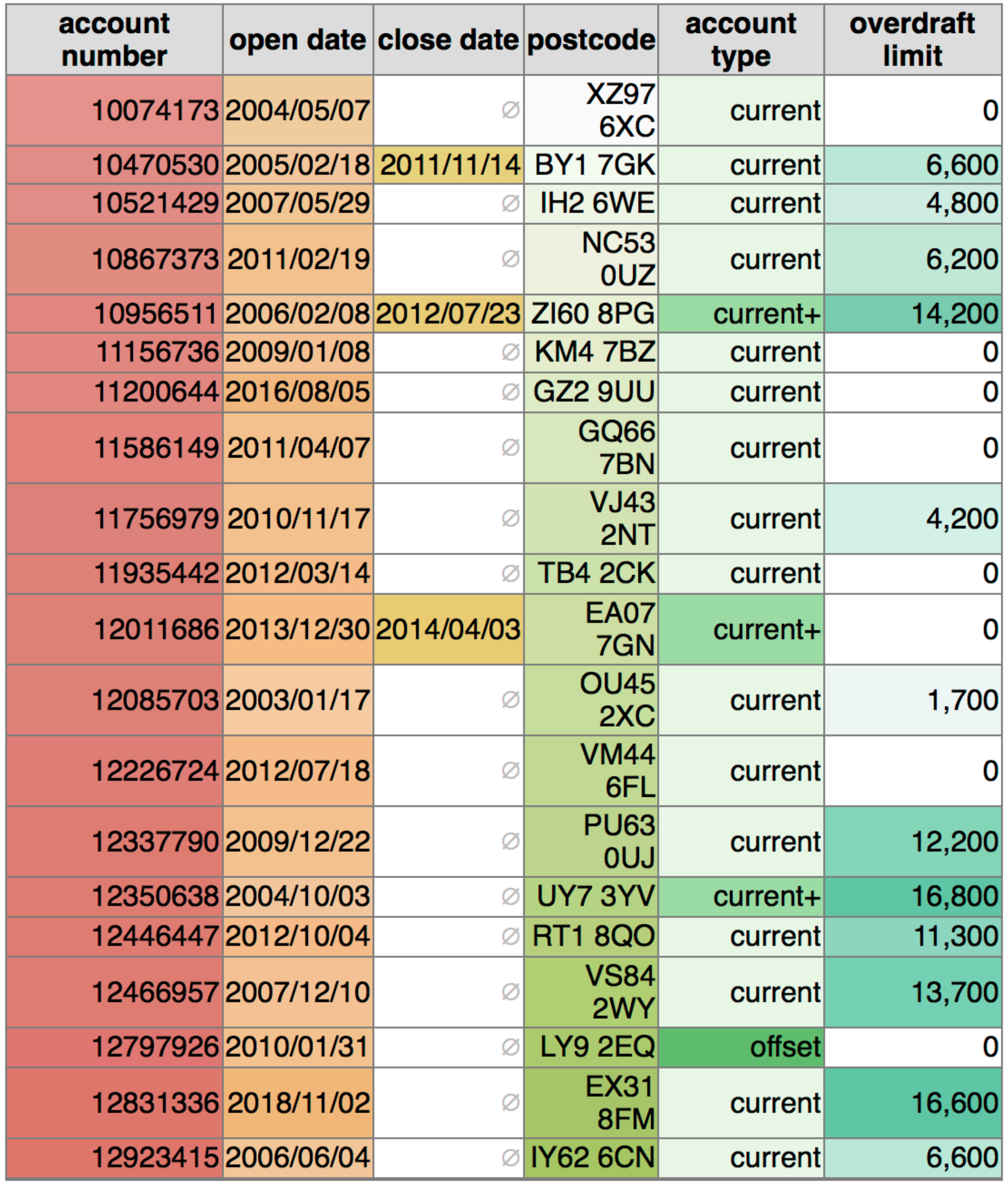
```
{
 C1: Age ≥ 0
 C2: ID is not null
 C3: CardNumber ~
       DDDD DDDD DDDD DDDD
             ⋮
}
```

DISCOVERED CONSTRAINTS

GENERATING CONSTRAINTS & VERIFYING DATA



EXAMPLE CONSTRAINT DISCOVERY



account number	open date	close date	postcode	account type	overdraft limit
10074173	2004/05/07	∅	XZ97 6XC	current	0
10470530	2005/02/18	2011/11/14	BY1 7GK	current	6,600
10521429	2007/05/29	∅	IH2 6WE	current	4,800
10867373	2011/02/19	∅	NC53 0UZ	current	6,200
10956511	2006/02/08	2012/07/23	ZI60 8PG	current+	14,200
11156736	2009/01/08	∅	KM4 7BZ	current	0
11200644	2016/08/05	∅	GZ2 9UU	current	0
11586149	2011/04/07	∅	GQ66 7BN	current	0
11756979	2010/11/17	∅	VJ43 2NT	current	4,200
11935442	2012/03/14	∅	TB4 2CK	current	0
12011686	2013/12/30	2014/04/03	EA07 7GN	current+	0
12085703	2003/01/17	∅	OU45 2XC	current	1,700
12226724	2012/07/18	∅	VM44 6FL	current	0
12337790	2009/12/22	∅	PU63 0UJ	current	12,200
12350638	2004/10/03	∅	UY7 3YV	current+	16,800
12446447	2012/10/04	∅	RT1 8QO	current	11,300
12466957	2007/12/10	∅	VS84 2WY	current	13,700
12797926	2010/01/31	∅	LY9 2EQ	offset	0
12831336	2018/11/02	∅	EX31 8FM	current	16,600
12923415	2006/06/04	∅	IY62 6CN	current	6,600

```
$ tdda discover -r training.csv constraints.tdda
{
    "creation_metadata": {
        "local_time": "2019-03-07 08:08:56",
        "utc_time": "2019-03-07 08:08:56",
        "creator": "TDDA 1.0.21",
        "source": "data.csv",
        "host": "bartok.local",
        "user": "njr",
        "dataset": "data.csv",
        "n_records": 20,
        "n_selected": 20,
        "tddafile": "constraints.tdda"
    },
    "fields": {
        .
        .
        .
```

EXAMPLE CONSTRAINT DISCOVERY

account number	open date	close date	postcode	account type	overdraft limit
10074173	2004/05/07	∅	XZ97 6XC	current	0
10470530	2005/02/18	2011/11/14	BY1 7GK	current	6,600
10521429	2007/05/29	∅	IH2 6WE	current	4,800
10867373	2011/02/19	∅	NC53 0UZ	current	6,200
10956511	2006/02/08	2012/07/23	ZI60 8PG	current+	14,200
11156736	2009/01/08	∅	KM4 7BZ	current	0
11200644	2016/08/05	∅	GZ2 9UU	current	0
11586149	2011/04/07	∅	GQ66 7BN	current	0
11756979	2010/11/17	∅	VJ43 2NT	current	4,200
11935442	2012/03/14	∅	TB4 2CK	current	0
12011686	2013/12/30	2014/04/03	EA07 7GN	current+	0
12085703	2003/01/17	∅	OU45 2XC	current	1,700
12226724	2012/07/18	∅	VM44 6FL	current	0
12337790	2009/12/22	∅	PU63 0UJ	current	12,200
12350638	2004/10/03	∅	UY7 3YV	current+	16,800
12446447	2012/10/04	∅	RT1 8QO	current	11,300
12466957	2007/12/10	∅	VS84 2WY	current	13,700
12797926	2010/01/31	∅	LY9 2EQ	offset	0
12831336	2018/11/02	∅	EX31 8FM	current	16,600
12923415	2006/06/04	∅	IY62 6CN	current	6,600

```
"account_number": {
    "type": "int",
    "min": 10074173,
    "max": 12923415,
    "sign": "positive",
    "max_nulls": 0,
    "no_duplicates": true
},
"open_date": {
    "type": "date",
    "min": "2003-01-17 00:00:00",
    "max": "2018-11-02 00:00:00",
    "max_nulls": 0
},
"close_date": {
    "type": "date",
    "min": "2011-11-14 00:00:00",
    "max": "2014-04-03 00:00:00"
},
```

EXAMPLE CONSTRAINT DISCOVERY

account number	open date	close date	postcode	account type	overdraft limit
10074173	2004/05/07	∅	XZ97 6XC	current	0
10470530	2005/02/18	2011/11/14	BY1 7GK	current	6,600
10521429	2007/05/29	∅	IH2 6WE	current	4,800
10867373	2011/02/19	∅	NC53 0UZ	current	6,200
10956511	2006/02/08	2012/07/23	ZI60 8PG	current+	14,200
11156736	2009/01/08	∅	KM4 7BZ	current	0
11200644	2016/08/05	∅	GZ2 9UU	current	0
11586149	2011/04/07	∅	GQ66 7BN	current	0
11756979	2010/11/17	∅	VJ43 2NT	current	4,200
11935442	2012/03/14	∅	TB4 2CK	current	0
12011686	2013/12/30	2014/04/03	EA07 7GN	current+	0
12085703	2003/01/17	∅	OU45 2XC	current	1,700
12226724	2012/07/18	∅	VM44 6FL	current	0
12337790	2009/12/22	∅	PU63 0UJ	current	12,200
12350638	2004/10/03	∅	UY7 3YV	current+	16,800
12446447	2012/10/04	∅	RT1 8QO	current	11,300
12466957	2007/12/10	∅	VS84 2WY	current	13,700
12797926	2010/01/31	∅	LY9 2EQ	offset	0
12831336	2018/11/02	∅	EX31 8FM	current	16,600
12923415	2006/06/04	∅	IY62 6CN	current	6,600

```
"postcode": {
    "type": "string",
    "min_length": 7,
    "max_length": 8,
    "max_nulls": 0,
    "no_duplicates": true,
    "rex": ["^[A-Z]{2}\\d{1,2} \\d[A-Z]{2}$"]
},
"account_type": {
    "type": "string",
    "min_length": 6,
    "max_length": 8,
    "max_nulls": 0,
    "allowed_values": [
            "current",
            "current+",
            "offset”
    ],
    "rex": ["^[a-z]{6,7}$", "^current\\+$"]
},
```

EXAMPLE CONSTRAINT DISCOVERY

account number	open date	close date	postcode	account type	overdraft limit
10074173	2004/05/07	∅	XZ97 6XC	current	0
10470530	2005/02/18	2011/11/14	BY1 7GK	current	6,600
10521429	2007/05/29	∅	IH2 6WE	current	4,800
10867373	2011/02/19	∅	NC53 0UZ	current	6,200
10956511	2006/02/08	2012/07/23	ZI60 8PG	current+	14,200
11156736	2009/01/08	∅	KM4 7BZ	current	0
11200644	2016/08/05	∅	GZ2 9UU	current	0
11586149	2011/04/07	∅	GQ66 7BN	current	0
11756979	2010/11/17	∅	VJ43 2NT	current	4,200
11935442	2012/03/14	∅	TB4 2CK	current	0
12011686	2013/12/30	2014/04/03	EA07 7GN	current+	0
12085703	2003/01/17	∅	OU45 2XC	current	1,700
12226724	2012/07/18	∅	VM44 6FL	current	0
12337790	2009/12/22	∅	PU63 0UJ	current	12,200
12350638	2004/10/03	∅	UY7 3YV	current+	16,800
12446447	2012/10/04	∅	RT1 8QO	current	11,300
12466957	2007/12/10	∅	VS84 2WY	current	13,700
12797926	2010/01/31	∅	LY9 2EQ	offset	0
12831336	2018/11/02	∅	EX31 8FM	current	16,600
12923415	2006/06/04	∅	IY62 6CN	current	6,600

```
            "overdraft_limit": {
                "type": "int",
                "min": 0,
                "max": 16800,
                "sign": "non-negative",
                "max_nulls": 0
            }
        }
    }
```

CONFIRM THAT CONSTRAINTS PASS ON TRAINING DATA

```
$ tdda verify training.csv constraints.tdda
account_number:  0 failures  6 passes
                 type ✓  min ✓  max ✓  sign ✓ max_nulls ✓  no_duplicates ✓
open_date:       0 failures  4 passes
                 type ✓  min ✓  max ✓  max_nulls ✓
close_date:      0 failures  3 passes
                 type ✓  min ✓  max ✓
postcode:        0 failures  6 passes
                 type ✓  min_length ✓  max_length ✓ max_nulls ✓
                 no_duplicates ✓  rex ✓
account_type:    0 failures  6 passes
                 type ✓  min_length ✓  max_length ✓ max_nulls ✓
                 allowed_values ✓  rex ✓
overdraft_limit: 0 failures  5 passes
                 type ✓  min ✓  max ✓  sign ✓  max_nulls ✓
Constraints passing: 30  Constraints failing: 0
```

CHECK WHETHER NEW DATA SATISFIES CONSTRAINTS

```
$ tdda verify operationaldata.csv constraints.tdda
account_number:  2 failures  4 passes
                 type ✓  min ✗  max ✗  sign ✓ max_nulls ✓  no_duplicates ✓
open_date:       1 failure  2 passes
                 type ✓  min ✗  max ✗  max_nulls ✓
close_date:      2 failures  1 pass
                 type ✓  min ✗  max ✗
postcode:        0 failures  6 passes
                 type ✓  min_length ✓  max_length ✓  max_nulls ✓
                 no_duplicates ✓  rex ✓
account_type:    3 failures  3 passes
                 type ✓  min_length ✗  max_length ✓ max_nulls ✓
                 allowed_values ✗  rex ✗
overdraft_limit: 1 failure  4 passes
                 type ✓  min ✓  max ✗  sign ✓  max_nulls ✓
Constraints passing: 21  Constraints failing: 9
```

FIND FAILING VALUES IN THE NEW DATA

```
$ tdda detect operationaldata.csv constraints.tdda failures.csv
account_number:  2 failures  4 passes
                 type ✓  min ✗  max ✗  sign ✓ max_nulls ✓  no_duplicates ✓
open_date:       1 failure  2 passes
                 type ✓  min ✗  max ✗  max_nulls ✓
close_date:      2 failures  1 pass
                 type ✓  min ✗  max ✗
postcode:        0 failures  6 passes
                 type ✓  min_length ✓  max_length ✓  max_nulls ✓
                 no_duplicates ✓  rex ✓
account_type:    3 failures  3 passes
                 type ✓  min_length ✗  max_length ✓ max_nulls ✓
                 allowed_values ✗  rex ✗
overdraft_limit: 1 failure  4 passes
                 type ✓  min ✓  max ✗  sign ✓  max_nulls ✓
Records passing: 76  Records failing: 24
```

account number	open date	close date	postcode	account type	overdraft limit	account number min ok	account number max ok	open date min ok	close date min ok	close date max ok	account type min ok	account type values ok	account type rex ok	overdraft limit max ok	nfailures
10033300	2005/02/08	∅	MO73 2YX	current	0	✗	✓	✓	∅	∅	✓	✓	✓	✓	**1**
10050552	2009/02/24	∅	XK5 3NM	current	0	✗	✓	✓	∅	∅	✓	✓	✓	✓	**1**
10066665	2003/02/16	∅	PI9 3BG	current+	0	✗	✓	✓	∅	∅	✓	✓	✓	✓	**1**
10174458	2011/07/18	**2016/09/27**	SX5 5PV	current	0	✓	✓	✓	✓	✗	✓	✓	✓	✓	**1**
10278760	2004/05/15	**2007/11/20**	BA72 8XF	current	**18,000**	✓	✓	✓	✗	✓	✓	✓	✓	✗	**2**
10352931	2004/06/15	∅	WJ9 2OA	**basic**	0	✓	✓	✓	∅	∅	✗	✗	✗	✓	**3**
10440004	**2002/12/19**	∅	YC24 4UT	current+	4,800	✓	✓	✗	∅	∅	✓	✓	✓	✓	**1**
10476972	2018/01/27	∅	OE5 9UI	current	**17,400**	✓	✓	✓	∅	∅	✓	✓	✓	✗	**1**
10699455	2018/09/17	∅	GQ1 9IV	current	**19,200**	✓	✓	✓	∅	∅	✓	✓	✓	✗	**1**
10717064	2003/11/30	∅	VM1 8WR	current	**20,000**	✓	✓	✓	∅	∅	✓	✓	✓	✗	**1**
10824167	2008/05/21	∅	NI55 0OS	**basic**	1,400	✓	✓	✓	∅	∅	✗	✗	✗	✓	**3**
10902721	2005/10/30	∅	LL22 5UX	current	**17,100**	✓	✓	✓	∅	∅	✓	✓	✓	✗	**1**
10962316	2003/12/25	**2005/02/25**	XX9 2RP	current	4,000	✓	✓	✓	✗	✓	✓	✓	✓	✓	**1**
11005672	2007/06/10	∅	ZT64 3WP	**basic**	0	✓	✓	✓	∅	∅	✗	✗	✗	✓	**3**
11385380	2015/08/07	∅	WC47 7OA	current+	**19,900**	✓	✓	✓	∅	∅	✓	✓	✓	✗	**1**
11589140	2007/11/04	∅	PF53 9BM	**basic**	8,300	✓	✓	✓	∅	∅	✗	✗	✗	✓	**3**
11604974	2008/04/27	**2010/02/18**	XE76 8YA	current	2,800	✓	✓	✓	✗	✓	✓	✓	✓	✓	**1**
11705553	2014/05/02	**2018/05/05**	LK55 9TE	current	0	✓	✓	✓	✓	✗	✓	✓	✓	✓	**1**
11816734	2012/04/27	∅	SS73 8VO	**basic**	15,200	✓	✓	✓	∅	∅	✗	✗	✗	✓	**3**
11957115	2007/04/01	∅	WO8 7QE	current	19,500	✓	✓	✓	∅	∅	✓	✓	✓	✗	**1**
12086022	2013/05/29	**2016/10/28**	UA06 1CI	**premium**	0	✓	✓	✓	✓	✗	✓	✗	✓	✓	**2**
12899220	2014/09/08	**2015/06/08**	UX80 2RO	current	0	✓	✓	✓	✓	✗	✓	✓	✓	✓	**1**
12940182	2017/12/13	∅	WA93 4SW	current	0	✓	✗	✓	∅	∅	✓	✓	✓	✓	**1**
12987964	2015/08/27	∅	SD83 3CR	current	0	✓	✗	✓	∅	∅	✓	✓	✓	✓	**1**

account number	open date	close date	postcode	account type	overdraft limit	account number min ok	account number max ok	open date min ok	close date min ok	close date max ok	account type min ok	account type values ok	account type rex ok	overdraft limit max ok	nfailures
10033300	2005/02/08	∅	MO73 2YX	current	0	✗	✓	✓	∅	∅	✓	✓	✓	✓	1
10050552	2009/02/24	∅	XK5 3NM	current	0	✗	✓	✓	∅	∅	✓	✓	✓	✓	1
10066665	2003/02/16	∅	PI9 3BG	current+	0	✗	✓	✓	∅	∅	✓	✓	✓	✓	1
10174458	2011/07/18	2016/09/27	SX5 5PV	current	0	✓	✓	✓	✓	✗	✓	✓	✓	✓	1
															2
															3
															1
															1
															1
															1
															3
															1
10962316	2003/12/25	2005/02/25	XX9 2RP	current	4,000	✓	✓	✓	✗	✓					1
11005672	2007/06/10	∅	ZT64 3WP	basic	0	✓	✓	✓	∅	∅					3
11385380	2015/08/07	∅	WC47 7OA	current+	19,900	✓	✓	✓	∅	∅					1
11589140	2007/11/04	∅	PF53 9BM	basic	8,300	✓	✓	✓	∅	∅					3
11604974	2008/04/27	2010/02/18	XE76 8YA	current	2,800	✓	✓	✓	✗	✓					1
11705553	2014/05/02	2018/05/05	LK55 9TE	current	0	✓	✓	✓	✓	✗	✓	✓	✓	✓	1
11816734	2012/04/27	∅	SS73 8VO	basic	15,200	✓	✓	✓	∅	∅	✗	✗	✗	✓	3
11957115	2007/04/01	∅	WO8 7QE	current	19,500	✓	✓	✓	∅	∅	✓	✓	✓	✗	1
12086022	2013/05/29	2016/10/28	UA06 1CI	premium	0	✓	✓	✓	✓	✗	✓	✗	✓	✓	2
12899220	2014/09/08	2015/06/08	UX80 2RO	current	0	✓	✓	✓	✓	✗	✓	✓	✓	✓	1
12940182	2017/12/13	∅	WA93 4SW	current	0	✓	✗	✓	∅	∅	✓	✓	✓	✓	1
12987964	2015/08/27	∅	SD83 3CR	current	0	✓	✗	✓	∅	∅	✓	✓	✓	✓	1

original data
for failing records

indicator columns
for each failing constraint

number of failures
for each record

account number	open date	close date	postcode	account type	overdraft limit	account number min ok	account number max ok	open date min ok	close date min ok	close date max ok	account type min ok	account type values ok	account type rex ok	overdraft limit max ok	nfailures
10033300	2005/02/08	∅	MO73 2YX	current	0	✗	✓	✓	∅	∅	✓	✓	✓	✓	1
10050552	2009/02/24	∅	XK5 3NM	current	0	✗	✓	✓	∅	∅	✓	✓	✓	✓	1
10066665	2003/02/16	∅	PI9 3BG	current+	0	✗	✓	✓	∅	∅	✓	✓	✓	✓	1
10174458	2011/07/18	2016/09/27	SX5 5PV	current	0	✓	✓	✓	✓	✗	✓	✓	✓	✓	1

account number
10033300
10050552
10066665

account number min ok
✗
✗
✗

nfailures
1
1
1

11005672	2007/06/10	∅	∅	✗	✗	✗	✓	3
11385380	2015/08/07	∅	∅	✓	✓	✓	✗	1
11589140	2007/11/04	∅	∅	✗	✗	✗	✓	3
11604974	2008/04/27	2010/02	✓	✓	✓	✓	✓	1
11705553	2014/05/02	2018/05	✗	✓	✓	✓	✓	1
11816734	2012/04/27	∅	∅	✗	✗	✗	✓	3
11957115	2007/04/01	∅	∅	✓	✓	✓	✗	1
12086022	2013/05/29	2016/10	✗	✓	✗	✓	✓	2
12899220	2014/09/08	2015/06	✗	✓	✓	✓	✓	1
12940182	2017/12/13	∅	∅	✓	✓	✓	✓	1
12987964	2015/08/27	∅	∅	✓	✓	✓	✓	1

```
"account_number": {
    "type": "int",
    "min": 10074173,
    "max": 12923415,
    "sign": "positive",
    "max_nulls": 0,
    "no_duplicates": true
},
```

Rexpy

Automatic construction of regular expressions from data

REGULAR EXPRESSIONS

```
212-988-0331
 476 123 8829
1 701 734 9288
(617) 222 0529
```

optional 1

```
^1?[ \(]?\d{3}\)?[ \-]\d{3}[ \-]\d{4}$
```

start of line	*optional space or open bracket*	*digits (3)*	*optional close bracket*	*space or hyphen*	*digits (3)*	*space or hyphen*	*digits (3)*	*end of line*

REGULAR EXPRESSIONS

212-977-0331

totally specific (overfitted)

```
^212\-977\-0331$
```

```
^[12]{3}\-[7-9]{3}\-(0|1|3){4}$
```

specific digits

```
^\d{3}\-\d{3}\-\d{4}$
```

What Rexpy produces

```
^\d+\-\d+\-\d+$
```

+ *means* ***"1 or more times"***

```
^.*$
```

. *matches any char*

* *means* ***"0 or more times"***

totally unspecific (underfitted)
(matches all strings)

REGULAR EXPRESSIONS

MN 55402

OH 45202

```
^[A-Z]{2} [0-9]{5}$
```

Some people, when confronted with a problem, think

"I know, I'll use regular expressions."

Now they have two problems.

— *Jamie Zawinski*
comp.emacs.xemacs, 1997

PROS

Powerful

Fast

Widely supported

CONS

*Ugly

Hard to write

Harder to read

Harder still to debug

Hard to quote/escape†

---

*Extremely . . .

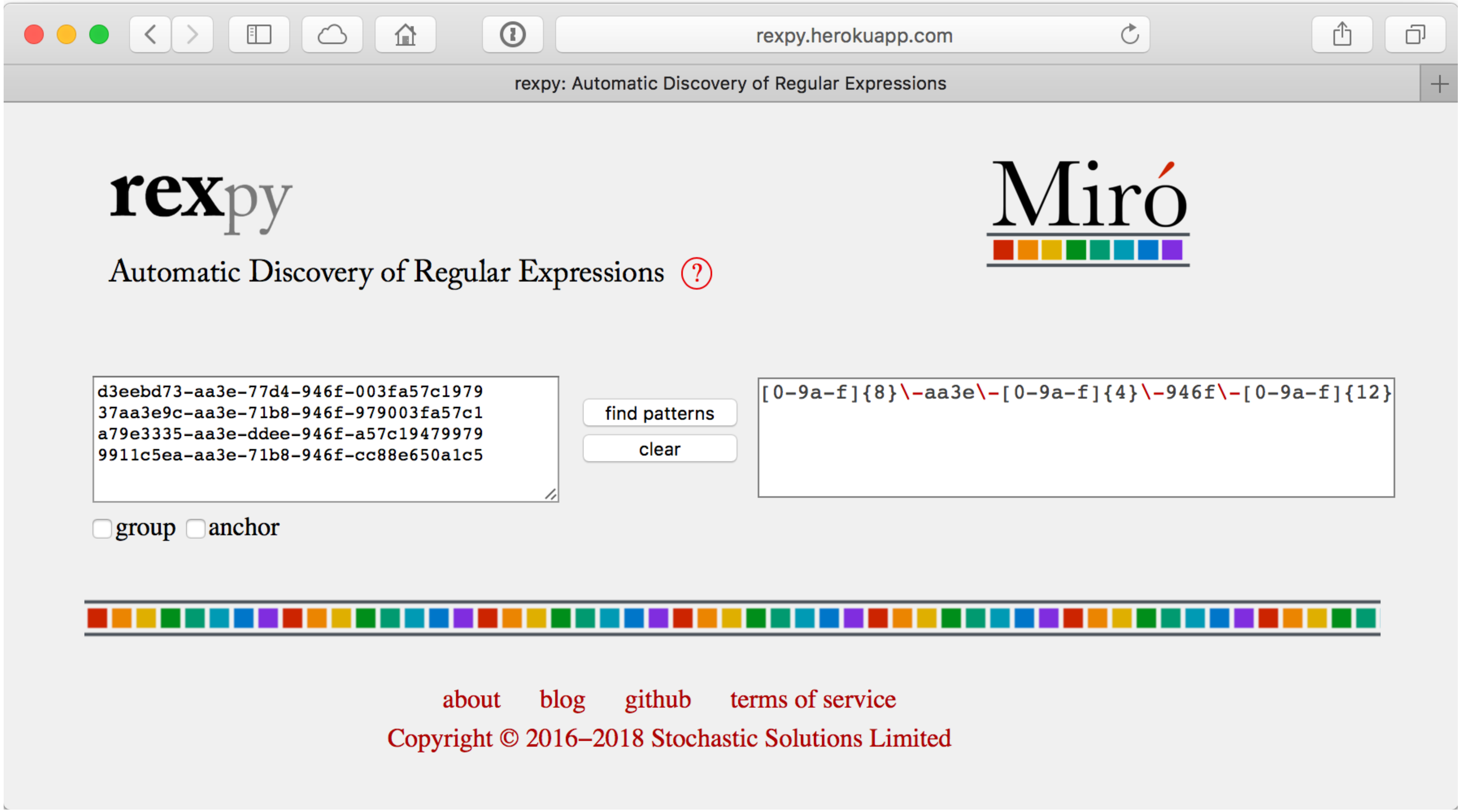
† `r'...'` is your friend

Why not let the computer do the work?

```
$ rexpy
212-988-0321
987-654-3210
476 123 8829
123 456 7890
701 734 9288
177 441 7712
```

```
^\d{3}\-\d{3}\-\d{4}$
^\d{3}\ \d{3}\ \d{4}$
```

Rexpy currently never groups white space with punctuation


rexpy.herokuapp.com
rexpy: Automatic Discovery of Regular Expressions
rexpy
Automatic Discovery of Regular Expressions
Miró
find patterns
clear
group
anchor
about
blog
github
terms of service
Copyright © 2016–2018 Stochastic Solutions Limited

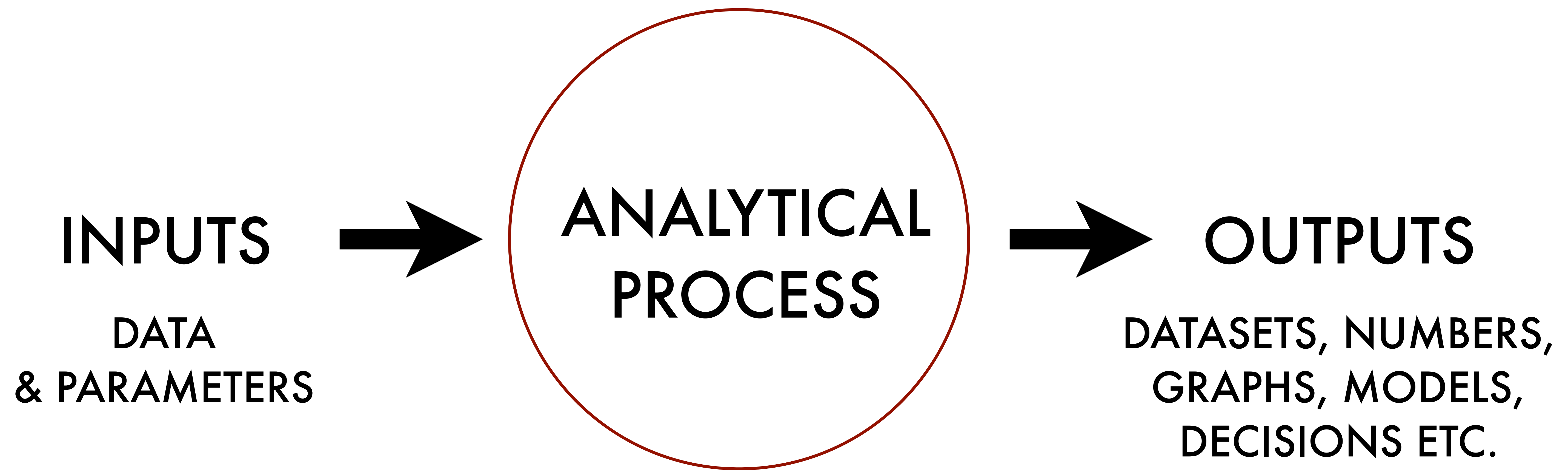
```
d3eebd73-aa3e-77d4-946f-003fa57c1979
37aa3e9c-aa3e-71b8-946f-979003fa57c1
a79e3335-aa3e-ddee-946f-a57c19479979
9911c5ea-aa3e-71b8-946f-cc88e650a1c5
```

```
[0-9a-f]{8}\-aa3e\-[0-9a-f]{4}\-946f\-[0-9a-f]{12}
```

rexpy.herokuapp.com

REFERENCE TESTS
&
AUTOMATIC TEST GENERATION
WITH TDDA GENTEST

REFERENCE TESTS



Record inputs

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

Record ("reference") outputs

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

REFERENCE TEST SUPPORT

1: UNSTRUCTURED (STRING) RESULTS

- Comparing actual string (in memory or in file) to reference (*expected*) string (in file)
- Exclude lines with substrings or regular expressions
- Preprocess output before comparison
- Write actual string produced to file when different
- Show specific diff command needed to examine differences
- Check multiple files in single test; report all failures
- Automatically re-write reference results after human verification.

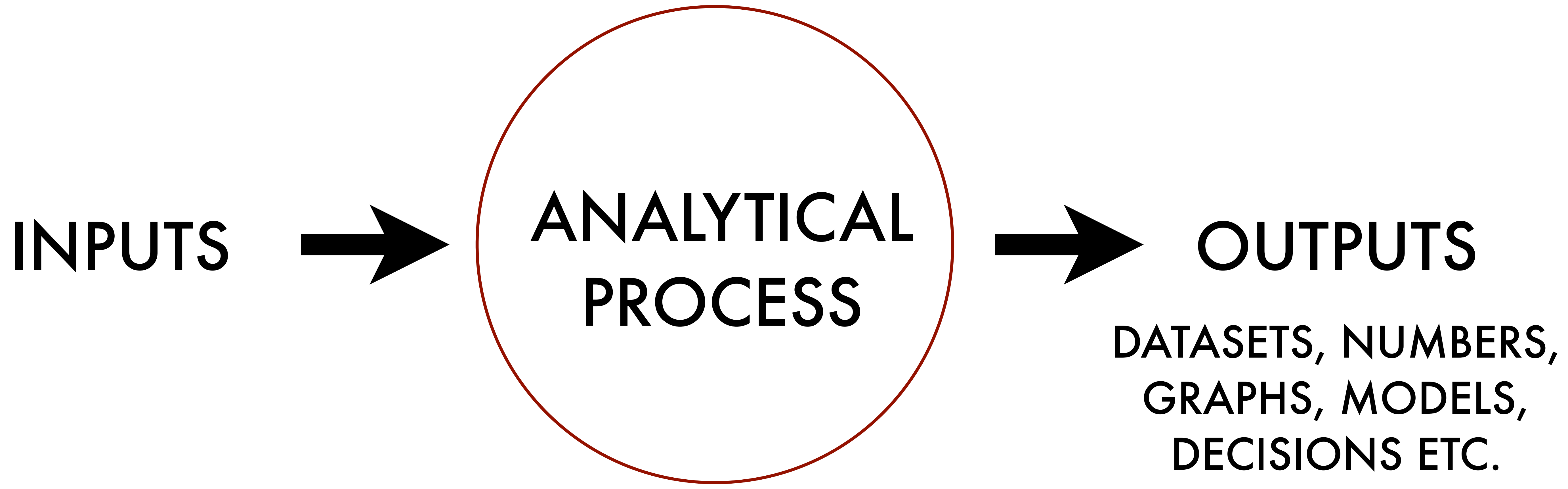
REFERENCE TEST SUPPORT

2: STRUCTURED DATA METHODS (DATAFRAMES & CSV)

- Comparing generated DataFrame or CSV file to reference DataFrame or CSV file
- Show specific diff command needed to examine differences
- Check multiple CSV files in single test; report all failures
- Choose subset of columns (with list or function) to compare
- Choose whether to check (detailed) types
- Choose whether to check column order
- Choose whether to ignore actual data in particular columns
- Choose precision for floating-point comparisons
- Automatic re-writing of verified (changed) results.

AUTOMATIC REFERENCE TESTS

coming soon!



Record inputs

Capture as script

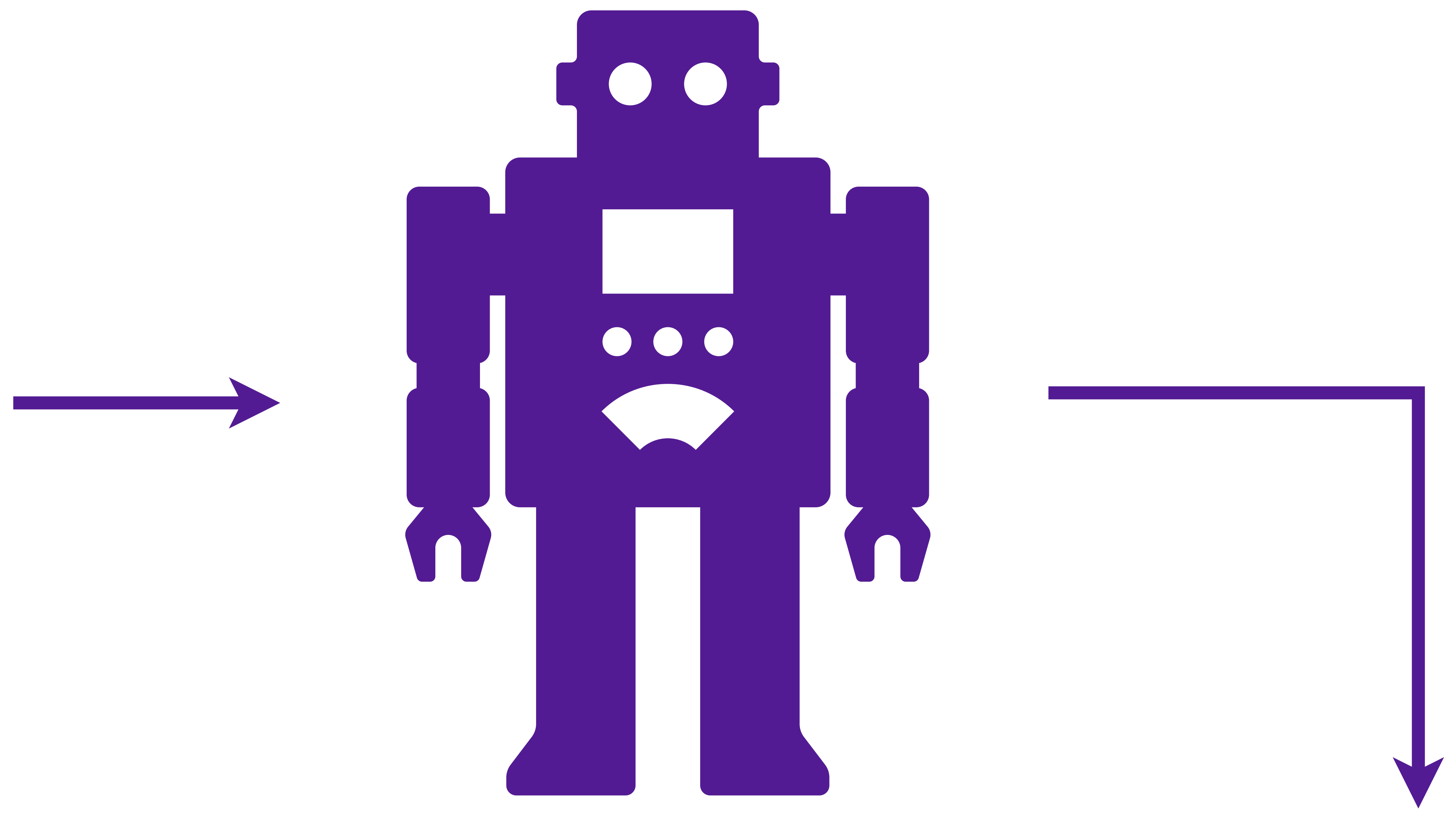
Record ("reference") outputs

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

GENTEST

```
tdda gentest "sh classify.sh"
```

```
sh classify.sh
```



test script	`test_sh_classify_sh.py`
reference outputs	`ref/sh_classify_sh`

Really?

GENTEST

stdout

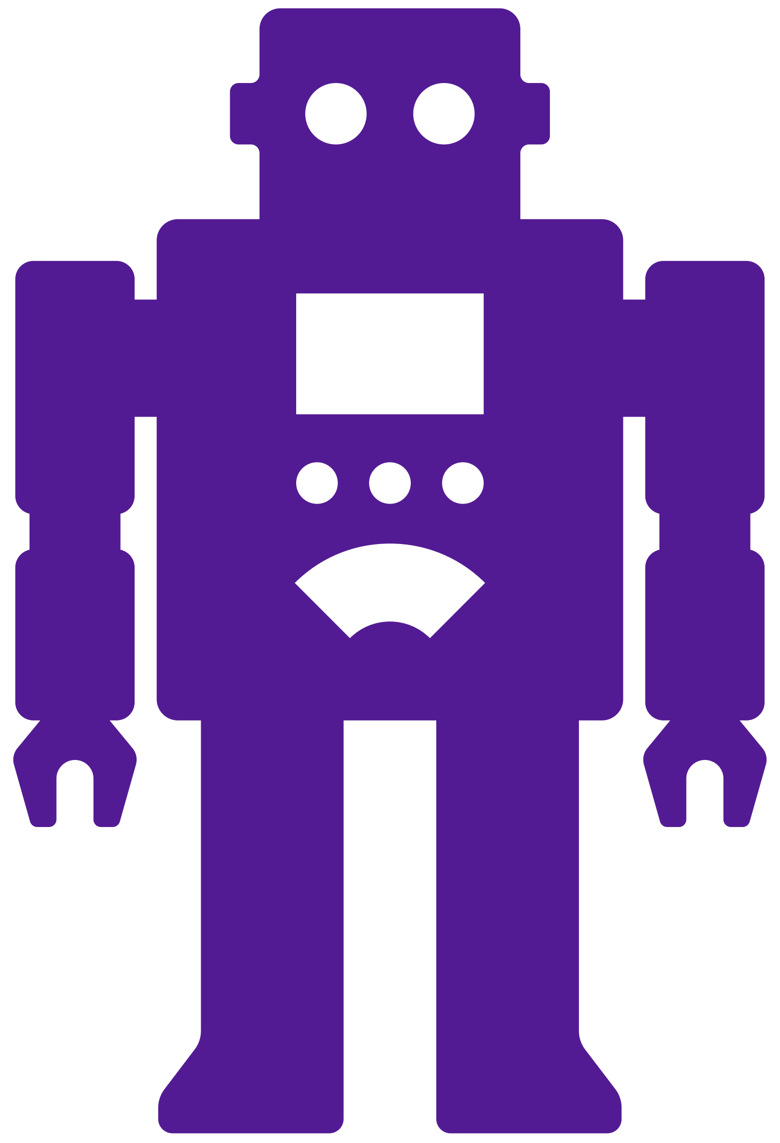
stderr

exit code

file system changes

environment (path, date, …)

differences between runs



↓

(artificially) intelligent decisions about how and what to test

Gentest is largely enabled by Rexpy!

MAR
20

DATA
FEST
2019

Testing Data & Data Processes with AI & Python

Wednesday, 20th March 2019, 14:00, Edinburgh

http://StochasticSolutions.com/training
http://www.datafest.global/fringe-events

http://stochasticsolutions.com/training

http://tdda.info

https://github.com/tdda

njr@StochasticSolutions.com

http://linkedin.com/in/njradcliffe

#tdda*

* *tweet (DM) us email address for invitation*
Or email me.

@tdda0 @njr0 @StochasticSolns

Correct interpretation: Zero (Error of interpretation: Letter "Oh")

http://stochasticsolutions.com/pdf/science-of-bad-data-datatech-2019.pdf