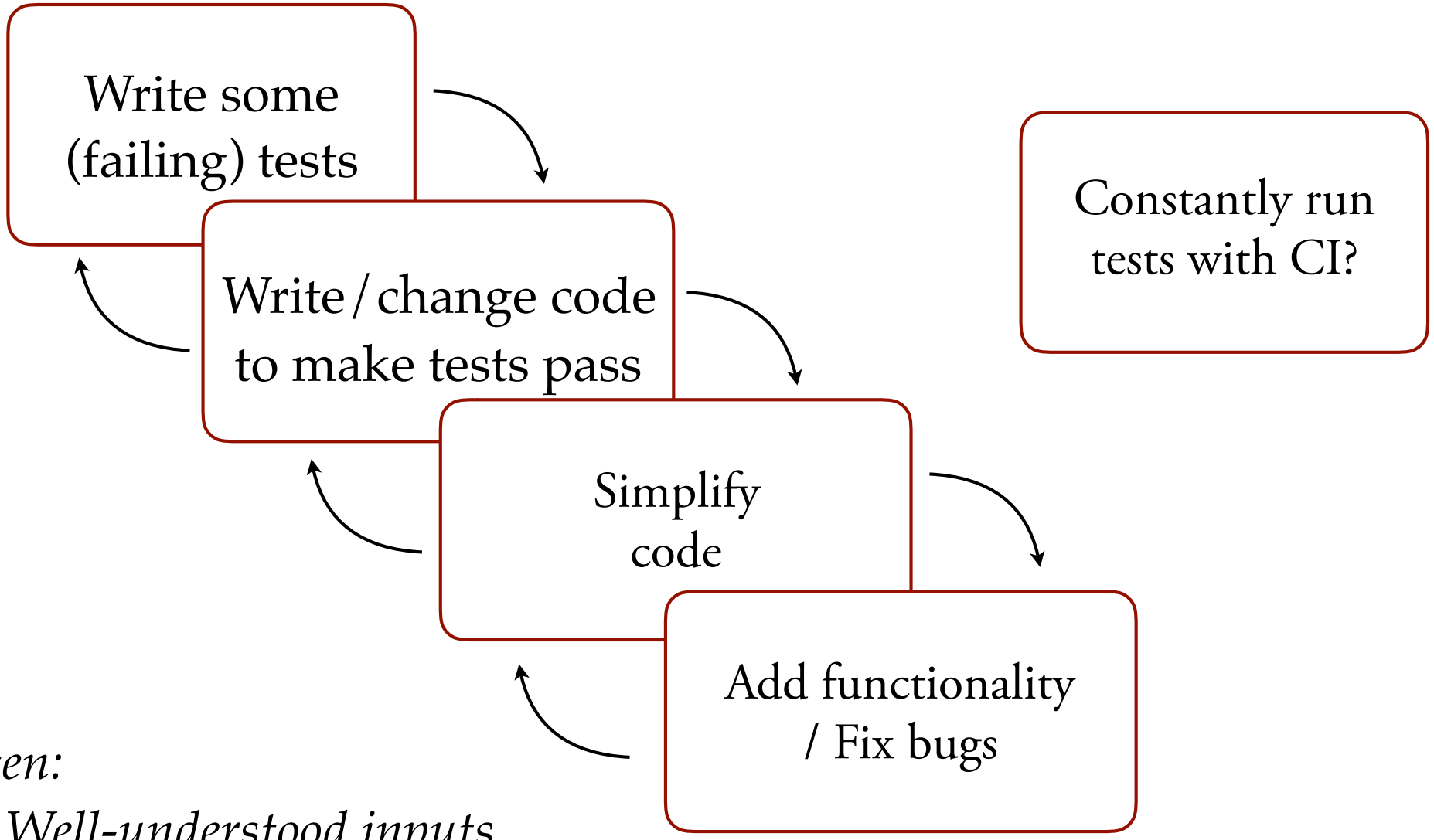
DEVELOPMENTS IN TEST-DRIVEN DATA ANALYSIS

PyData Berlin 2017 • 1st July 2017

http://www.tdda.info/pdf/tdda-pydata-berlin-2017.pdf

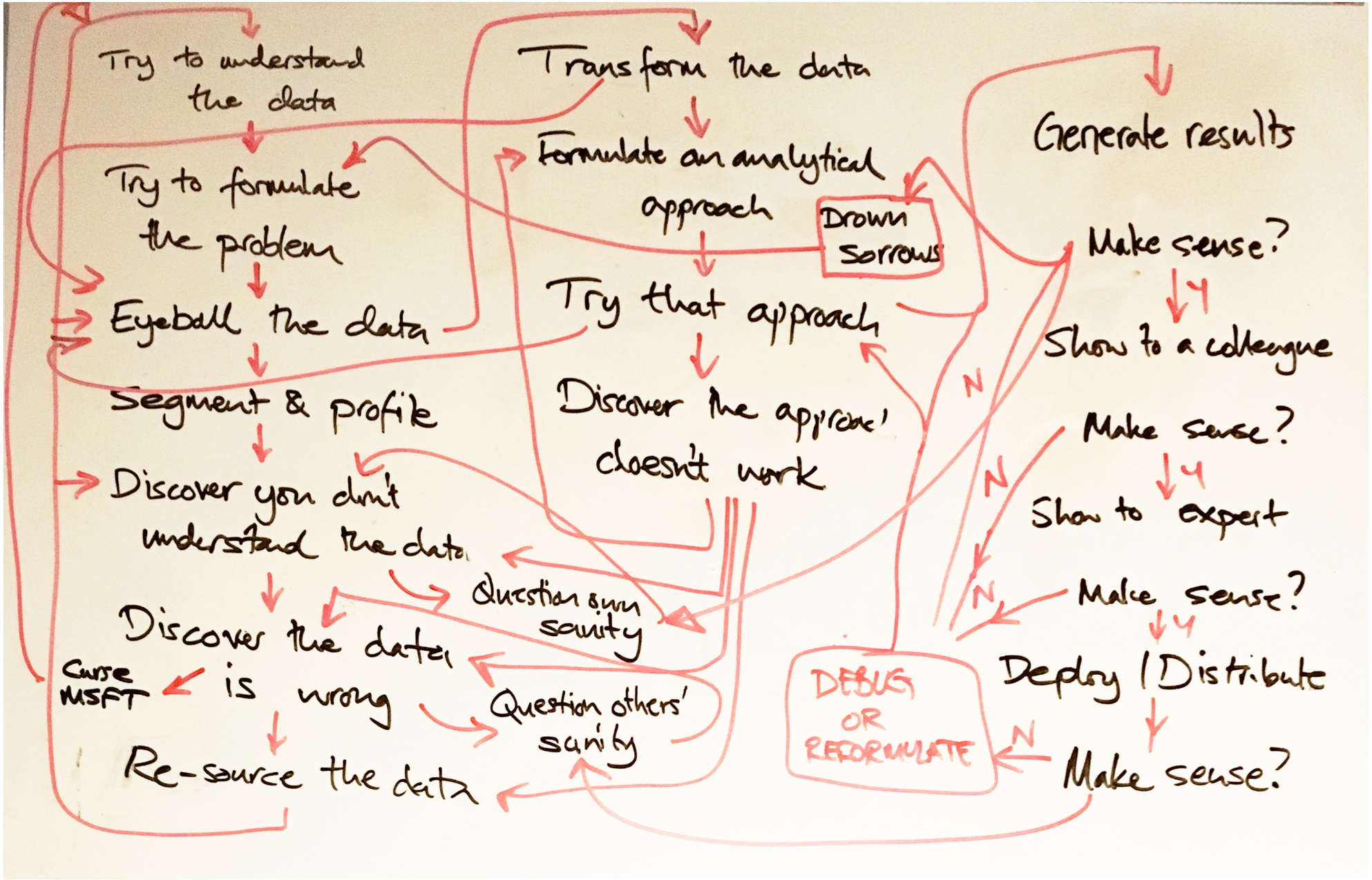
Nick Radcliffe
Stochastic Solutions Limited
& Department of Mathematics, University of Edinburgh

SOFTWARE DEVELOPMENT (WITH TDD)



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
Transform the data
Formulate an analytical approach
Drown sorrows
Try that approach
Discover the approach doesn't work
Question own sanity
Question others' sanity
DEBUG OR REFORMULATE
Generate results
Make sense?
Y
Show to a colleague
Make sense?
Y
Show to expert
Make sense?
Y
Deploy / Distribute
Make sense?
N
N
N
N

TDD ↦ TDDA

We need to extend TDD's idea of testing for

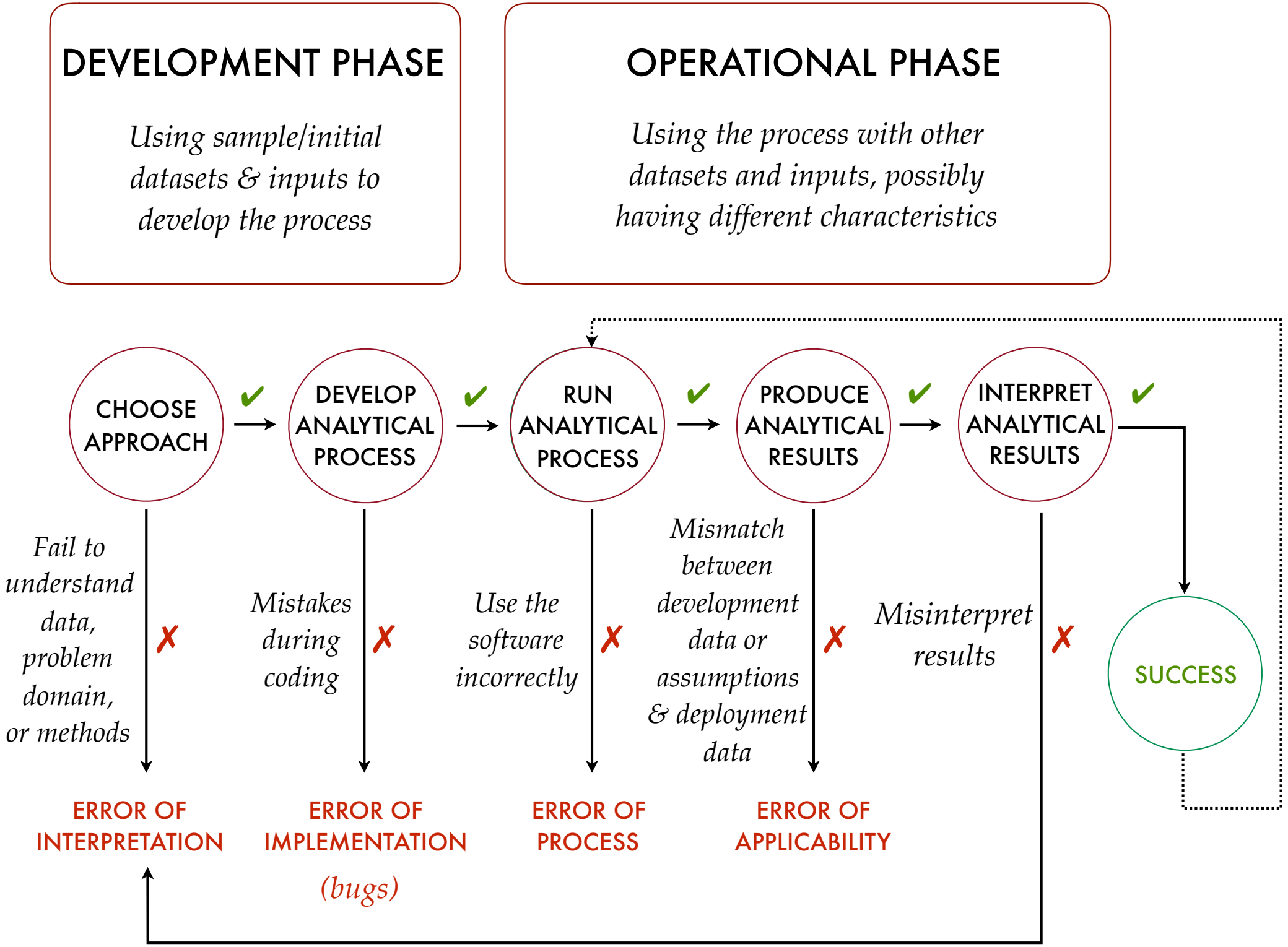
software correctness

with the idea of testing for

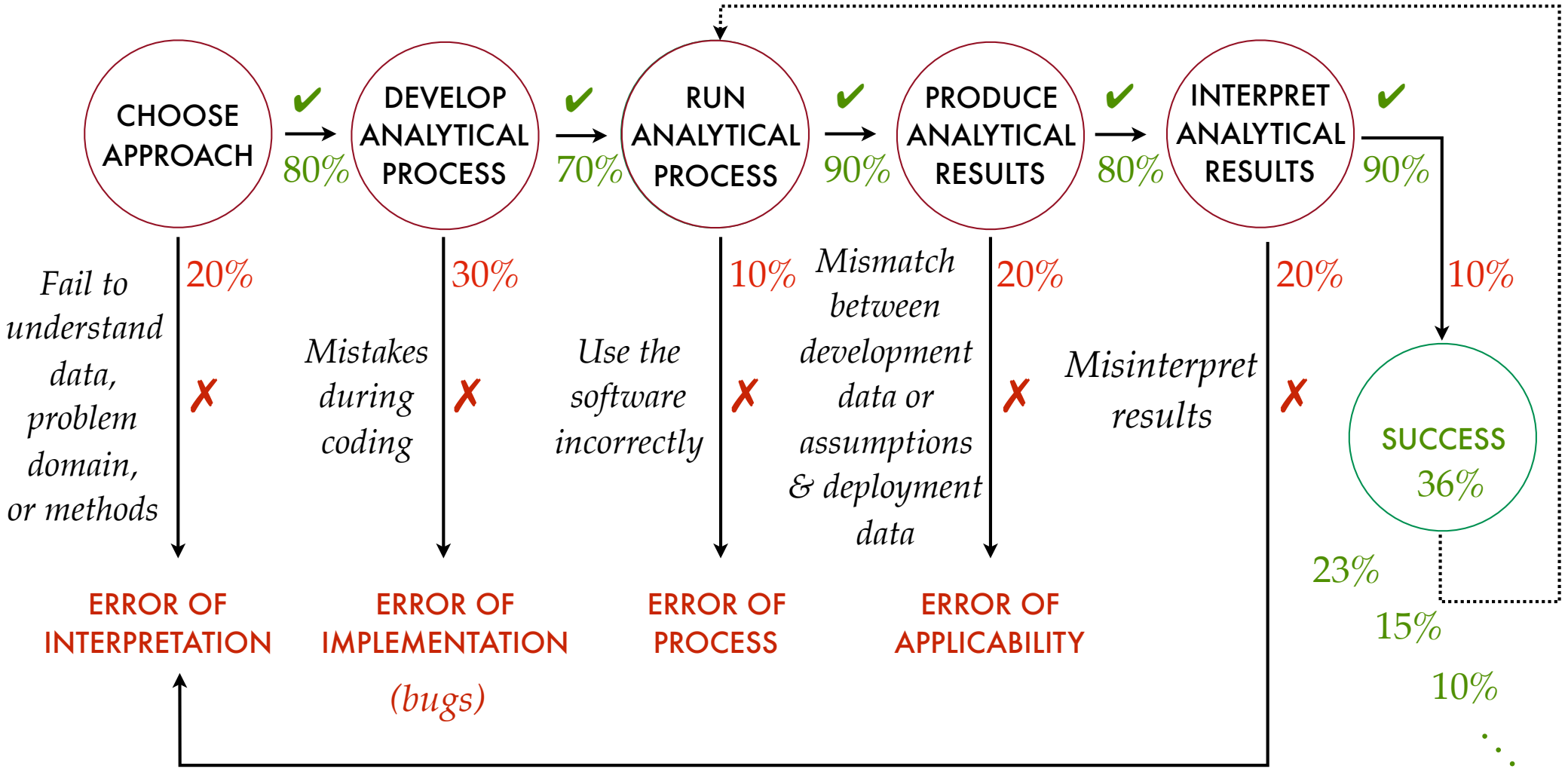
meaningfulness of analysis,

correctness and validity of input data,

& *correctness of interpretation*.


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 . . .



TDDA: MAIN IDEAS

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

TDDA LIBRARY

1. From PyPI (recommended)

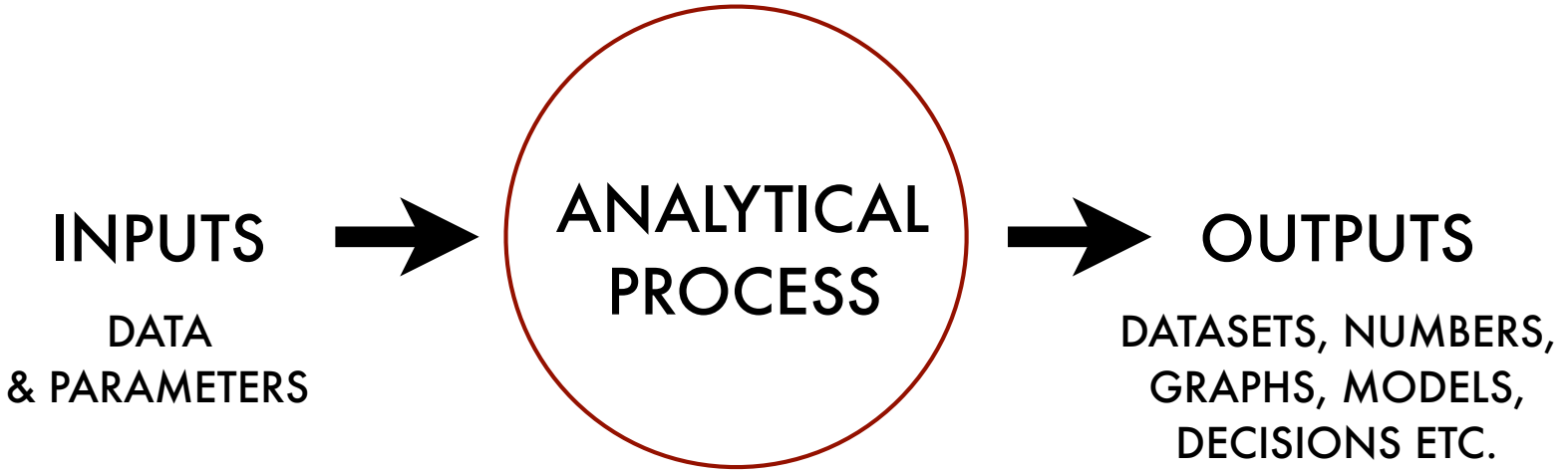
   ```
   pip install tdda
   ```

2. From Github (source)

   ```
   git clone https://github.com/tdda/tdda.git
   ```

- 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://pythonhosted.org/tdda`
- Quick reference: `http://www.tdda.info/pdf/tdda-quickref.pdf`

REFERENCE TESTS



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

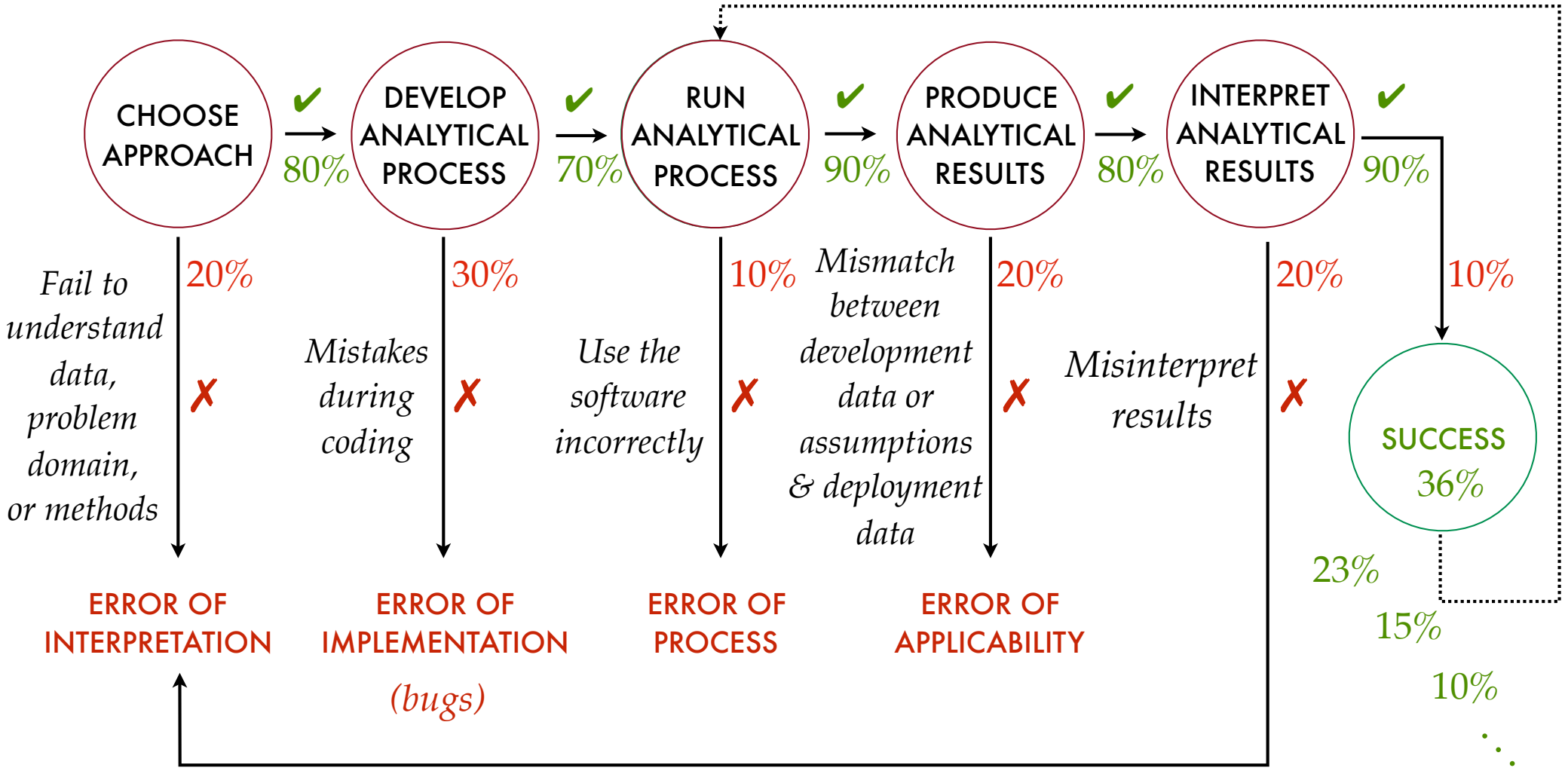
2: STRUCTURED DATA METHODS (DATAFRAMES & CSV)

- 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.

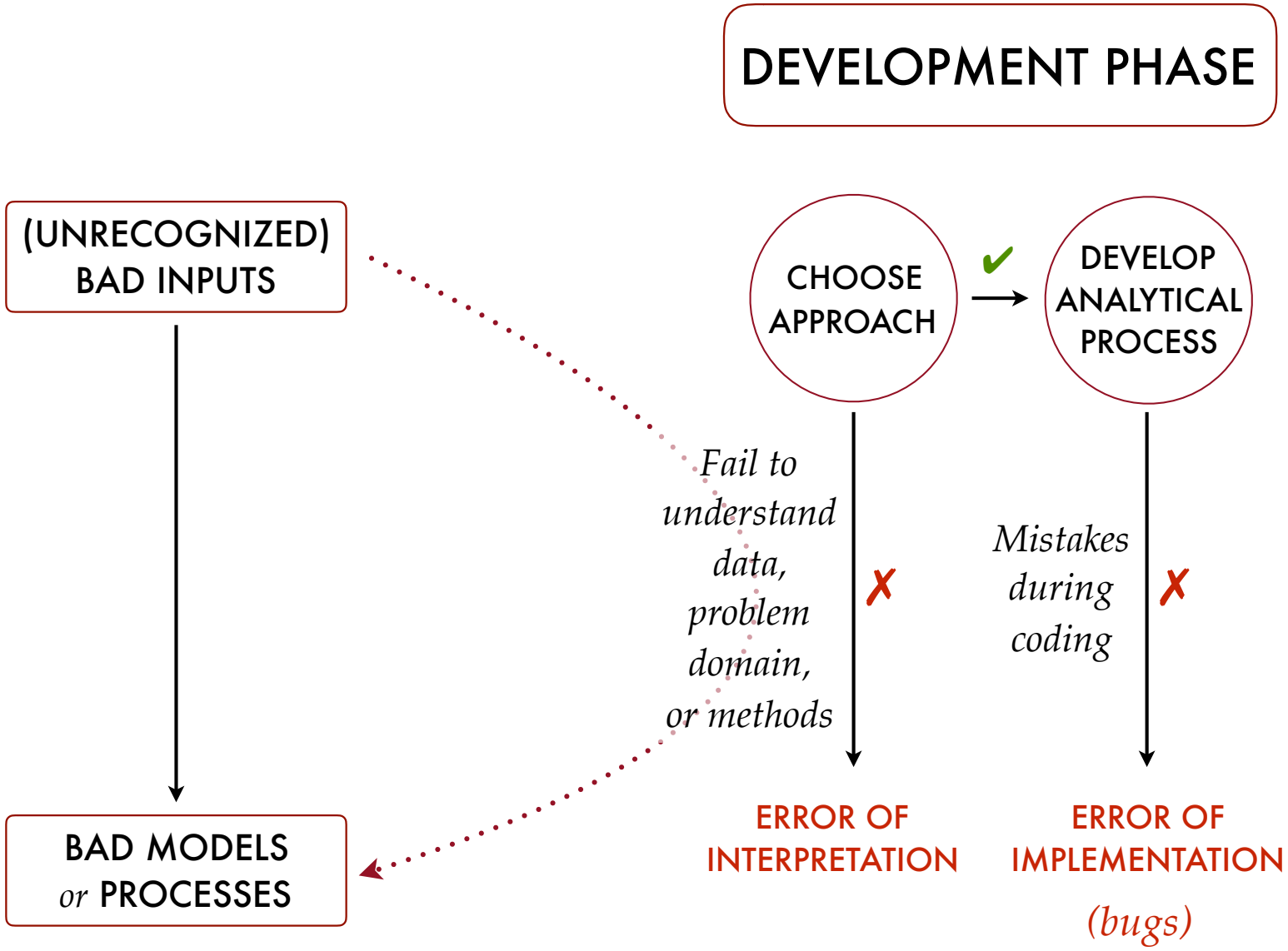
demonstration:
reference tests

CONSTRAINT GENERATION & VERIFICATION

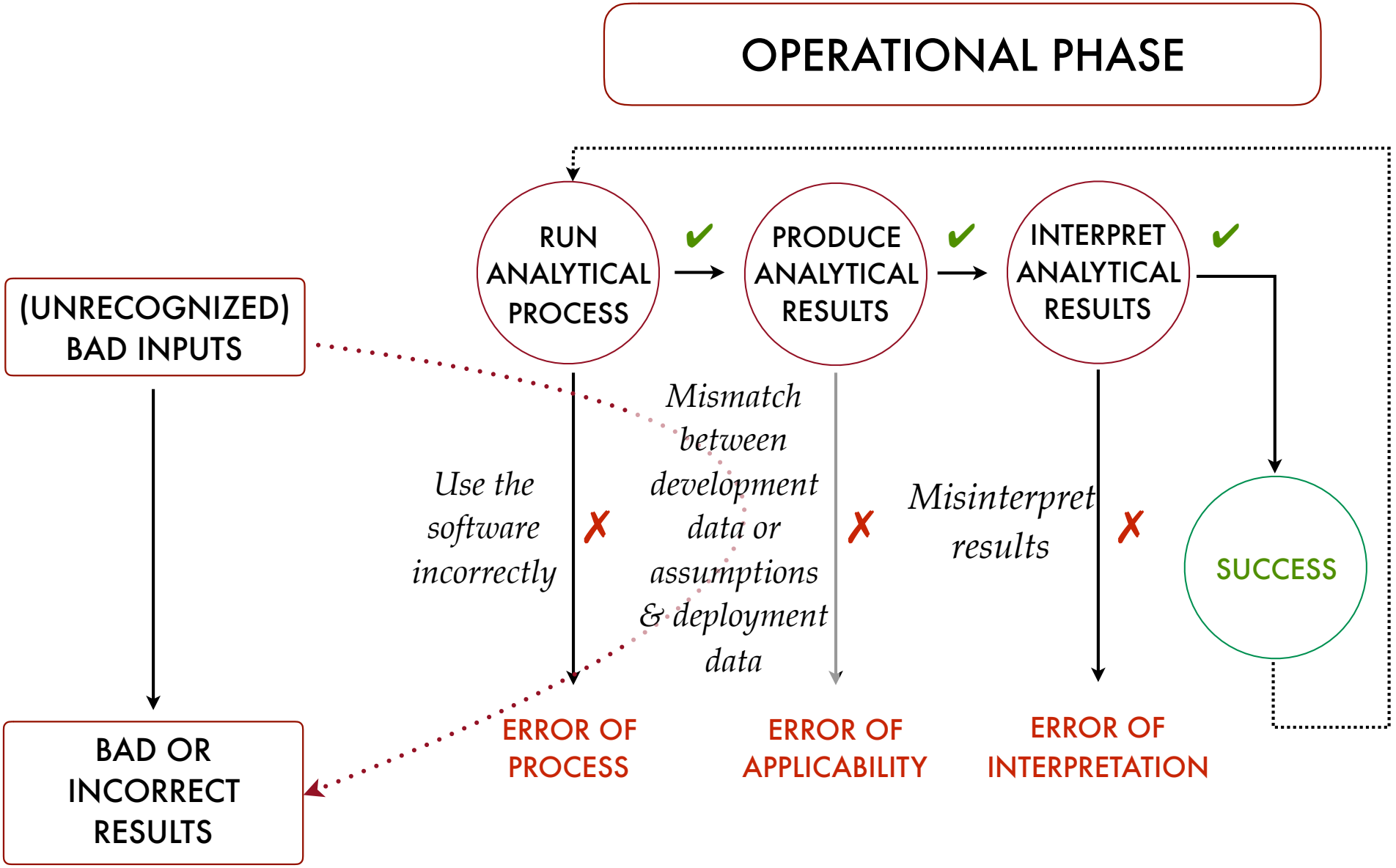
HOW DOES OUR PROCESS HOLD UP IF THERE IS BAD DATA?



BAD DATA DURING DEVELOPMENT



BAD DATA DURING OPERATION



CONSTRAINTS

- Very commonly, data analysis uses data tables (e.g. DataFrames) 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)

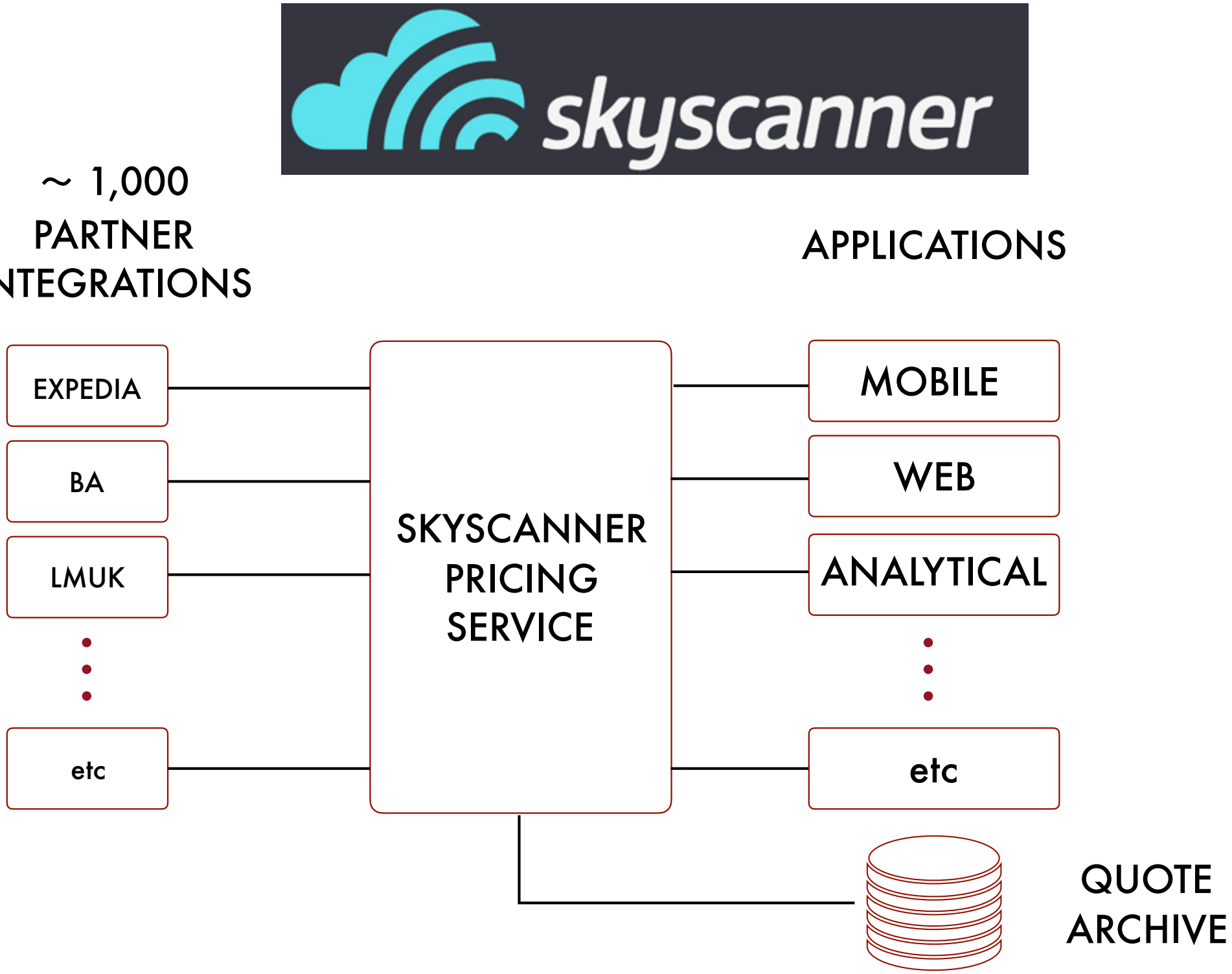
THE BIG IDEA

- 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)

EXAMPLE CONSTRAINTS

SINGLE FIELD CONSTRAINTS	DATASET CONSTRAINTS
Age ≤ 150	The dataset must contain field **CID**
type(Age) = int	Number of records must be 118
CID ≠ NULL	One field should be tagged O
CID unique	Date should be sorted ascending
len(CardNumber) = 16	**MULTI-FIELD CONSTRAINTS**
Base in {“C”, “G”, “A”, T”}	StartDate ≤ EndDate
NI ~ ^[A-Z]{2} \d{2} \d{2} \d{2} [A-Z]$	AlmostEqual(F, m * a, 6)
StartDate < tomorrow()	sum(Favourite*) = 1
v < 2.97e8	minVal ≤ medianVal ≤ maxVal
Height ~ N(1.8, 0.2)	V ≤ H * w * d

EXAMPLE


skyscanner
~ 1,000
PARTNER
INTEGRATIONS
APPLICATIONS
EXPEDIA
BA
LMUK
etc
SKYSCANNER
PRICING
SERVICE
MOBILE
WEB
ANALYTICAL
etc
QUOTE
ARCHIVE

demonstration: constraint generation & validation

SAMPLE DATA

Request Id	Page Request Date	Platform Name	Website ID	Website	Category	User Country ID	User Lang	Redirect From Code	Redirect From Country Code	Redirect To Code
082ddfbc-16f4-11e5-8664-49ac424641a9	2015/06/20 02:28:27	website	kbns	rdtgwtbphk	trava	RU	ru	DME	RU	AZN
c7c5cc0e-171c-11e5-b65e-b912e4e0d1a8	2015/06/20 07:20:08	website	fwtg	ggxsklrqcff	trava	AU	en	SYD	AU	ATH
04b1ecd0-177f-11e5-877d-eb4455a0cdcb	2015/06/20 19:03:21	website	bfuk	ooulrrux	trava	RO	ro	OTP	RO	BKK
d6a33e80-1732-11e5-a789-5b7713b	2015/06/20 09:58:02	website	bqwe	bbzcp	airl	SE	sv	ARN	SE	LHR
12084542-176f-11e5-8360-55f81437ffc1	2015/06/20 17:09:11	ipad	ruet	gjxftfzordpnfb	trava	IT	it	MXP	IT	MIA

SAMPLE DATA ctd

Number Of Adults	Number Of Children	Number Of Infants	Airline ID	Cabin Class	Price Currency	Ticket Price GBP	Card Price GBP	Total Price GBP	Card
1	0	0	HY	economy	RUB	106.70	∅	106.70	MASTERCARD DEBIT
4	0	0	60	economy	AUD	5,080.55	342.17	4,984.80	MASTERCARD DEBIT
1	0	0	SU	economy	EUR	391.18	7.83	399.56	MASTERCARD CREDIT
1	0	0	AY	business	SEK	779.55	0.00	779.55	AMEX
1	0	0	UX	economy	EUR	626.51	0.00	639.43	VISA DEBIT

AUTO-GENERATED CONSTRAINTS

Individual Field Constraints								
Name	Type	Min	Max	Sign	Max Nulls	Dups	Values	# regex
RequestId	string	length 36	length 36		0			1
PageRequestDate	date	2015/06/20 01:42:23	2016/01/20 23:51:16		0			
:time-before-now	timedelta	526 days, 20:30:29	741 days, 18:39:22	> 0				
PlatformName	string	length 4	length 13		0		6 values	1
WebsiteID	string	length 4	length 4		0			1
Website	string	length 3	length 24		0			1
Category	string	length 5	length 5		0		2 values	1
UserCountryID	string	length 2	length 2		0			1
UserLang	string	length 2	length 2		0			1
RedirectFromCode	string	length 3	length 3		0			1
RedirectFromCountryCode	string	length 2	length 2		0			1

AUTO-GENERATED CONSTRAINTS

Individual Field Constraints								
Name	Type	Min	Max	Sign	Max Nulls	Dups	Values	# regex
RedirectToCountryCode	string	length 2	length 2		0			1
NumberOfAdults	int	1	8	> 0				
NumberOfChildren	int	0	5	≥ 0				
NumberOfInfants	int	0	2	≥ 0				
AirlineID	string	length 2	length 2		0			1
CabinClass	string	length 5	length 15		0		4 values	2
PriceCurrency	string	length 3	length 3		0			1
TicketPrice_GBP	real	6.62	18,397.88	> 0	0			
CardPrice_GBP	real	-119,140.79	587,862.19					
TotalPrice_GBP	real	0.00	14,193,537.43	≥ 0	0			
Card	string	length 3	length 48		0			8

AUTO-GENERATED CONSTRAINTS

Individual Field Constraints								
Name	Type	Min	Max	Sign	Max Nulls	Dups	Values	# regex
RedirectToCountryCode	string	length 2	length 2		0			1
NumberOfAdults	int	1	8	> 0				
NumberOfChildren	int	0	5	≥ 0				
NumberOfInfants	int	0	2	≥ 0				
AirlineID	string	length 2	length 2		0			1
CabinClass	string	length 5	length 15		0		4 values	2
PriceCurrency	string	length 3	length 3		0			1
TicketPrice_GBP	real	6.62	18,397.88	> 0	0			
CardPrice_GBP	real	-119,140.79	587,862.19					
TotalPrice_GBP	real	0.00	14,193,537.43	≥ 0	0			
Card	string	length 3	length 48		0			8

ABSENT CONSTRAINTS

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.

EXTRACT FROM TDDA FILE


```
"fields": {
   "RequestId": {
      "type": "string",
      "min_length": 36,
      "max_length": 36,
      "max_nulls": 0,
      "rex": [
         "^([0-9a-f]{8})\\-([0-9a-f]{4})\\-11e5\\-([0-9a-f]{4})\\-([0-9a-f]{12})$"
      ]
   },
   "PageRequestDate": {
      "type": "date",
      "min": "2015/06/20 01:42:23",
      "max": "2016/01/20 23:51:16",
      "max_nulls": 0
   },
```

VERIFICATION 1

Name	Failures	Type			Minimum			Maximum		
		Allowed	Actual	✓	Allowed	Actual	✓	Allowed	Actual	✓
PageRequestDate	2	date	date	✓	2015/06/20 01:42:23	2015/06/19 23:05:05	✗	2016/01/20 23:51:16	2016/01/20 23:57:13	✗
TicketPrice_GBP	2	real	real	✓	6.62	0.01	✗	18,397.88	72,082.60	✗
CardPrice_GBP	2	real	real	✓	-119,140.79	-4,900,329.70	✗	587,862.19	877,213.03	✗
PageRequestDate:time-before-now	1	-	-	-	526 days, 19:52:36	527 days, 12:31:26	✓	741 days, 18:01:29	742 days, 13:23:34	✗
NumberOfChildren	1	int	int	✓	0	0	✓	5	8	✗
NumberOfInfants	1	int	int	✓	0	0	✓	2	3	✗
TotalPrice_GBP	1	real	real	✓	0.00	0.00	✓	14,193,537.43	20,242,428.5	✗
Card	1	string	string	✓	length 3	length 3	✓	length 48	length 48	✓

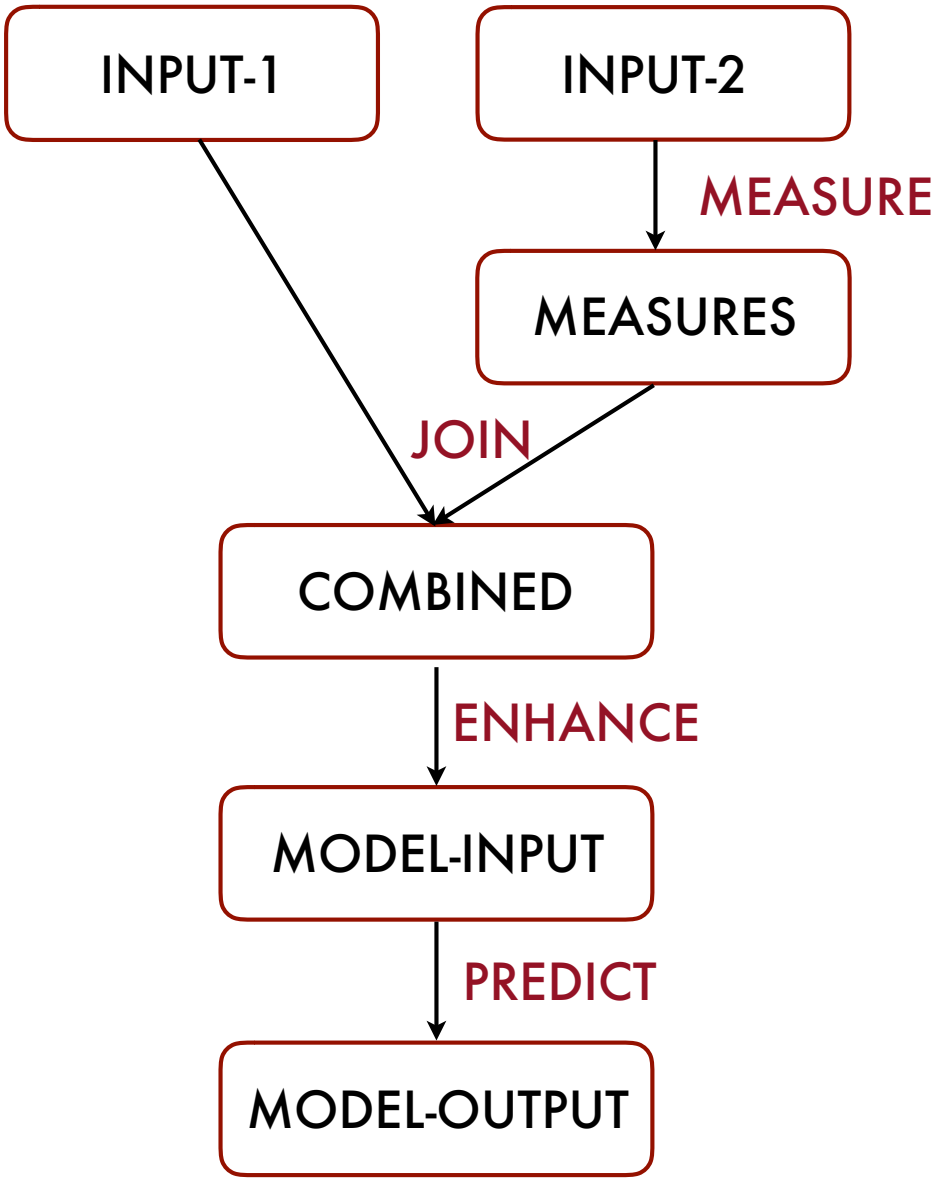
VERIFICATION 2

Name	Sign			Max Nulls			Duplicates		
	Allowed	Actual	✓	Allowed	Actual	✓	Allowed	Actual	✓
PageRequestDate	-	-	-	0	0	✓	-	-	-
TicketPrice_GBP	> 0	✓	✓	0	0	✓	-	-	-
CardPrice_GBP	-	-	-	-	38646	-	-	-	-
PageRequestDate:time-before-now	> 0	✓	✓	-	0	-	-	-	-
NumberOfChildren	≥ 0	✓	✓	-	68	-	-	-	-
NumberOfInfants	≥ 0	✓	✓	-	68	-	-	-	-
TotalPrice_GBP	≥ 0	✓	✓	0	0	✓	-	-	-
Card	-	-	-	0	0	✓	-	-	-

VERIFICATION 3

Name	Values			Rex		
	Allowed	Actual	✓	Allowed	Actual	✓
PageRequestDate	-	-	-	-	-	-
TicketPrice_GBP	-	-	-	-	-	-
CardPrice_GBP	-	-	-	-	-	-
PageRequestDate:time-before-now	-	-	-	-	-	-
NumberOfChildren	-	-	-	-	-	-
NumberOfInfants	-	-	-	-	-	-
TotalPrice_GBP	-	-	-	-	-	-
Card	-	-	-	8 patterns	e.g. "BANK OF BAHRAIN AND	✗

INTERMEDIATES & OUTPUTS TOO



input1
.tdda

input2
.tdda

measures
.tdda

combined
.tdda

model-input
.tdda

model-output
.tdda

Rexpy

Regular Expressions by Example

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

REGULAR EXPRESSIONS

212-988-0321

476 123 8829

1 701 734 9288

(617) 222 0529

optional 1

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

start of line /string | *optional space or open bracket* | *digits (3)* | *optional close bracket* | *space or hyphen* | *digits (3)* | *space or hyphen* | *digits (4)* | *end of line /string*

REGULAR EXPRESSIONS

MN 55402

OH 45202-7735

^[A–Z]{2} \d{5}(\–\d{4})?$

unescaped parentheses (no backslash) "tag" sub-expressions

optional

PROS

Powerful

CONS

Ugly

Hard to write

Harder to read

Harder still to debug

Hard to quote/escape†

---

† `r'...'` is your friend

Why not let
the computer do
the work?

demonstration

(rexpy)

Rexpy

is our early attempt to let the computer
find useful regular expressions
from examples

included in tdda library
(but not yet used to generate/check constraints)
and also available online at rexpy.herokuapp.com

TDDA FUTURES

TDDA CURRENT

Item	Difficulty	Used *ad hoc*	In Miró	In tdda lib
Reference tests	medium	✔	✔	✔
Type constraints	easy	✔	✔	✔
Min/Max constraints (inc. lengths)	easy	✔	✔	✔
Sign constraints	easy	✔	✔	✔
Nulls constraints	easy	✔	✔	✔
Duplicates constraints	easy	✔	✔	✔
Categorical values constraints	medium	✔	✔	✔
Regular expression generation	hard	✔	✔	✔
Regular expression constraints	easy	✔	✔	*real soon . . .*
Time delta (relative time) constraints	medium	✔	✔	*planned*
Compare pairs of date fields	easy	✔	✔	*planned*
Foreign key constraints	medium	✔	½	*planned*

TDDA FUTURES 1

- Combining/updating constraints sets
  - ★ Narrowing and broadening
- Conditional constraints
  - ★ *e.g. only for data after 2017-07-01; only for Berlin data*
- Incident characterization/Root cause analysis
- Reporting of many kinds, inc. grouping failing values
  - ★ Visualization
- More field generation e.g. virtual fields from regex groups
- Encapsulate identities/checks with columns
- JSON and JSON schemas

TDDA FUTURES 2

- Distribution shifts
- “Nearly” constraints
- Rexpy improvements
- Constraints editor
- Constraint severity levels
- Alerting
- Human feedback (inc. errors!)
- Different kinds of data feeds

REMEMBER YOUR ABC

Always

Be

Checking

njr@StochasticSolutions.com

http://tdda.info

https://github.com/tdda

#tdda*

@tdda0 @njr0

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

Correct interpretation: Zero

Error of interpretation: Letter "Oh"

www.tdda.info/pdf/tdda-pydata-berlin-2017.pdf