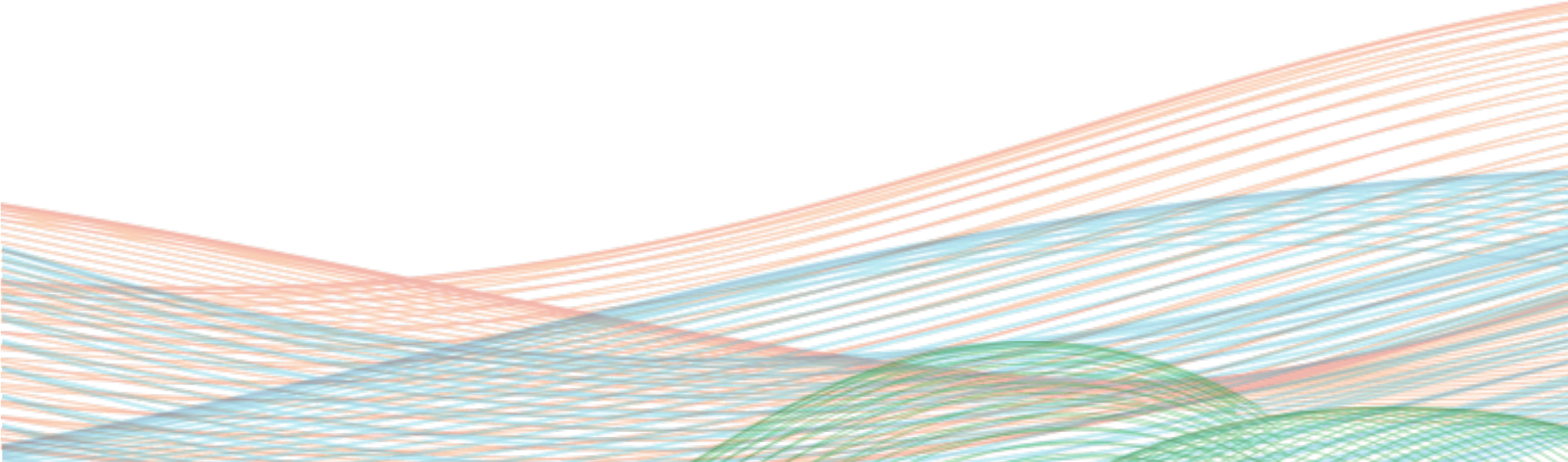


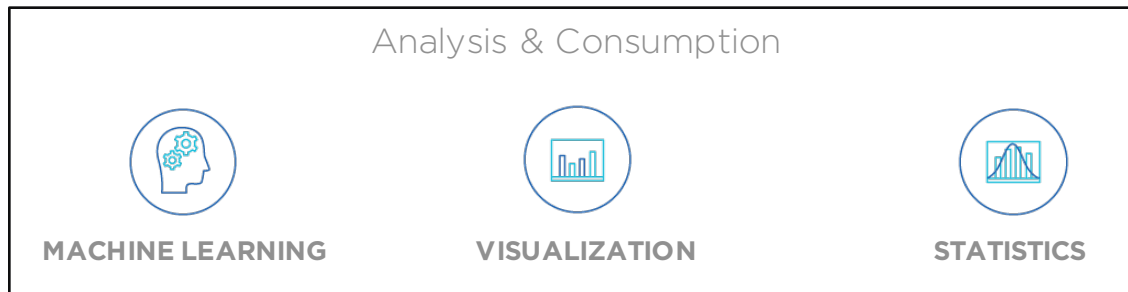
An abstract graphic at the bottom of the slide consists of numerous thin, overlapping lines in shades of orange, blue, and green, creating a sense of depth and movement.

Amelia Arbisser
Adam Silberstein

Scalable and Incremental
Data Profiling with Spark

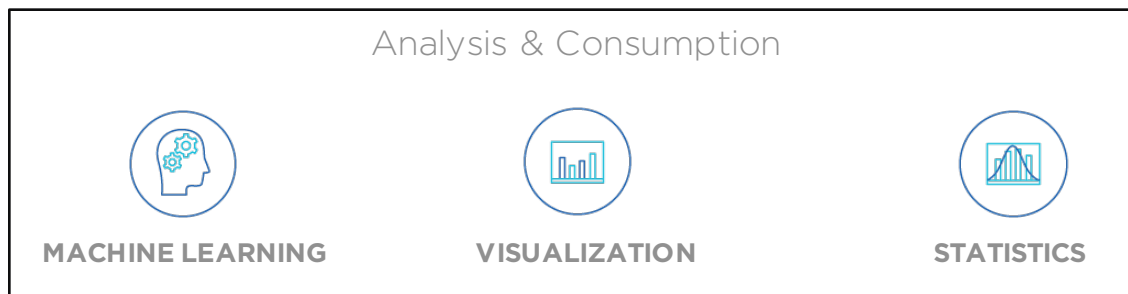
Trifacta: Self-service data preparation

What data analysts hope to achieve in data projects



Trifacta: Self-service data preparation

What data analysts hope to achieve in data projects

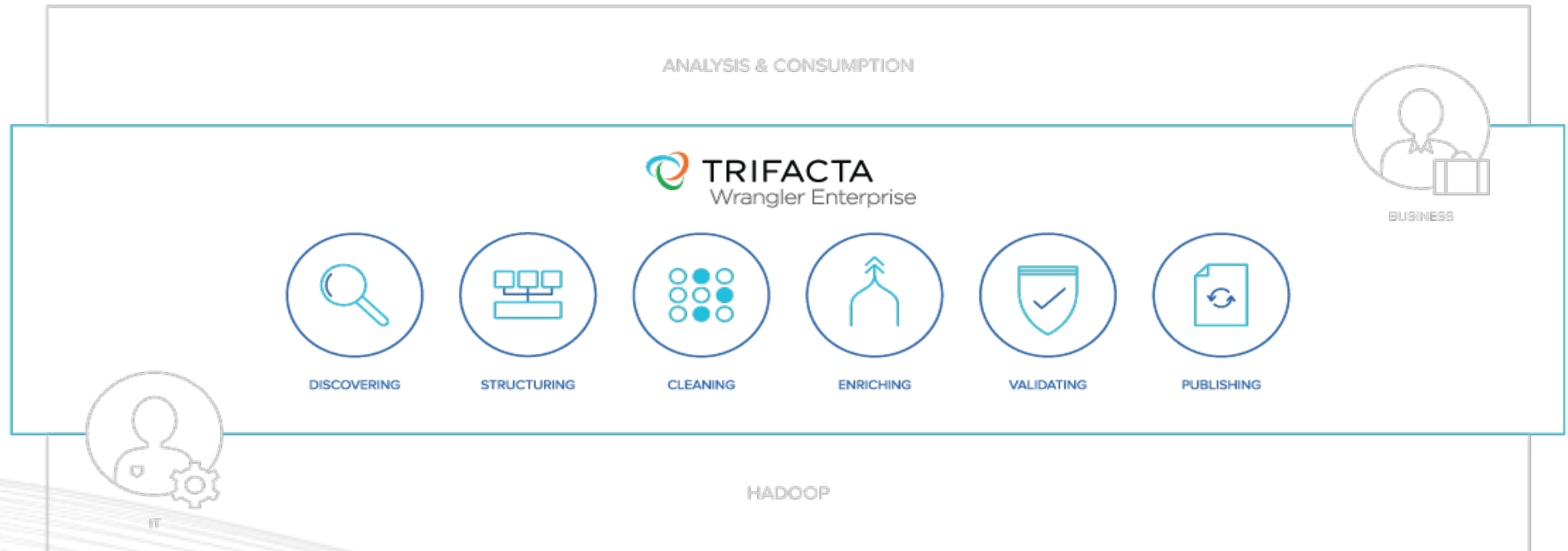


80% of time spent cleaning and preparing the data to be analyzed



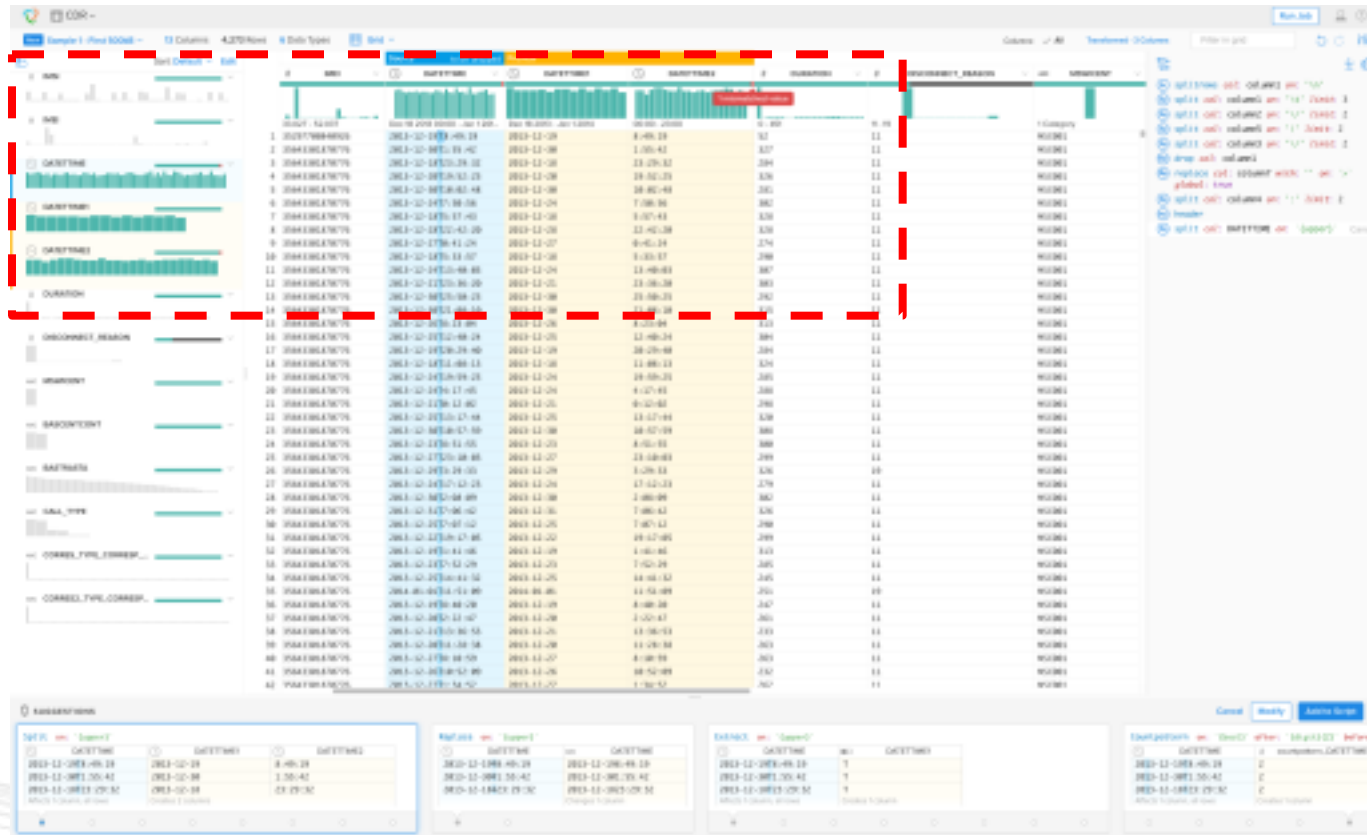
Trifacta: Self-service data preparation

We speed up and take the pain out of preparation

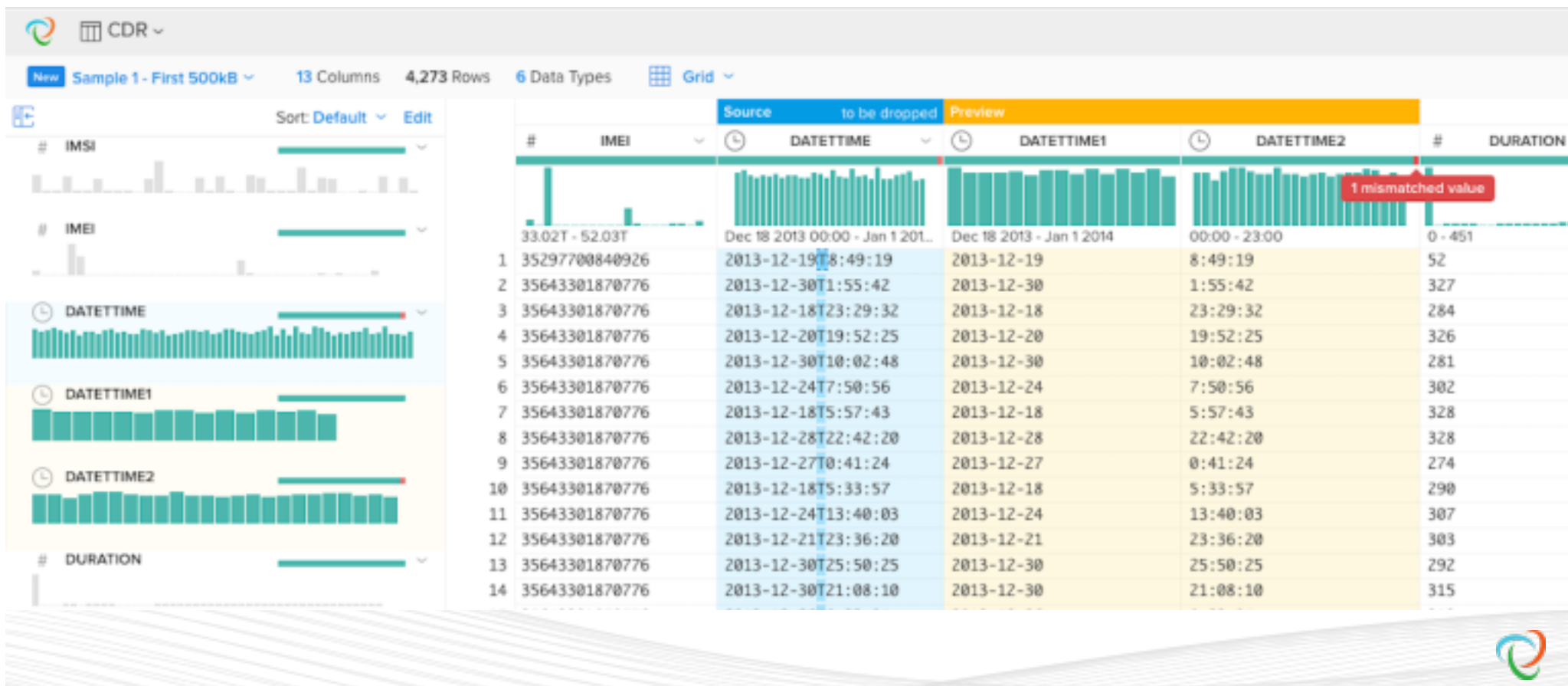




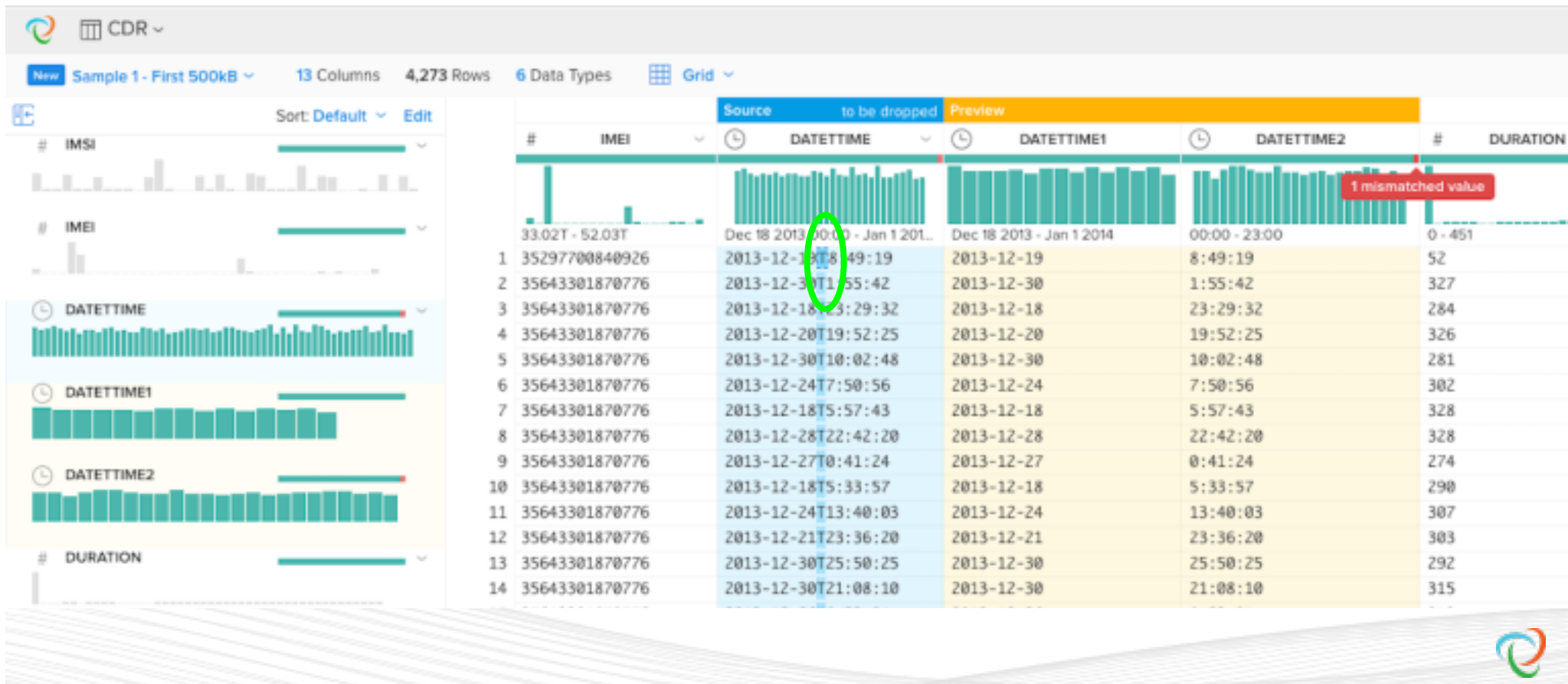
Transforming Data in Trifacta



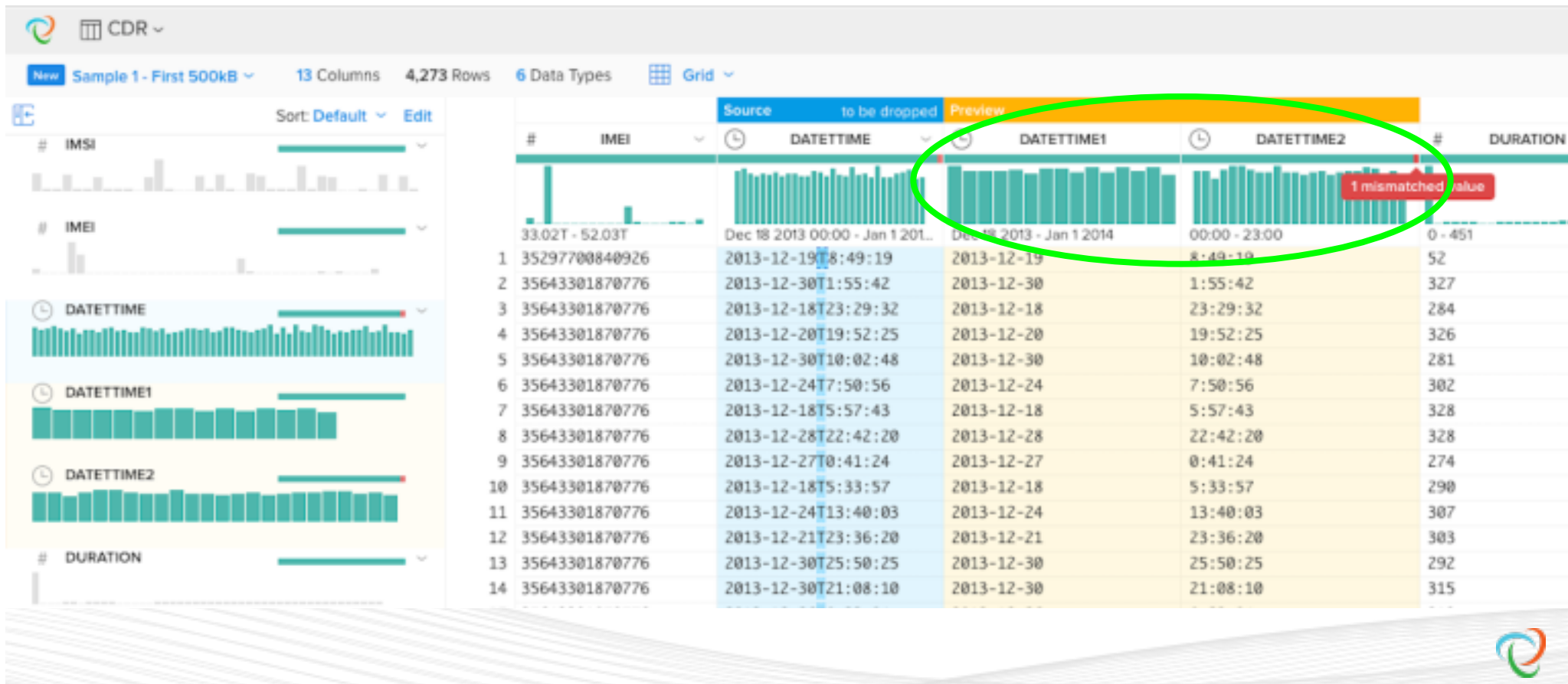
Predictive Interaction and Immediate Feedback



Predictive Interaction



Immediate Feedback







DURATION	#	DISCONNECT_REASON	ABC	MSWICENT	ABC	BASCENTCONT
	11 - 19		1 Category		2 Categories	
	11		MSC001		BSC001	
	11		MSC001		BSC001	
	11		MSC001		BSC002	
	11		MSC001		BSC002	
	11		MSC001		BSC002	
	11		MSC001		BSC002	
	11		MSC001		BSC001	
	11		MSC001		BSC002	
	11		MSC001		BSC001	
	11		MSC001		BSC001	
	11		MSC001		BSC001	
	11		MSC001		BSC001	

⚙️ ⬇️ 🤖
 Ⓢ splitrows col: column1 on: '\n'
 Ⓢ split col: column1 on: '\t' limit: 3
 Ⓢ split col: column2 on: '\v' limit: 2
 Ⓢ split col: column4 on: ':' limit: 2
 Ⓡ replace col: column7 with: '' on: '>' global: true
 Ⓡ replace col: column1 with: '' on: '<' global: true
 Ⓢ split col: column3 on: '\v' limit: 2
 Ⓜ header
 Ⓢ split col: DATETIME on: '{upper}' Cancel

Cancel

Modify

Add to Script

Split on: '{upper}'

DATETIME	DATETIME1	DATETIME2
2013-12-19T8:49:19	2013-12-19	8:49:19
2013-12-30T1:55:42	2013-12-30	1:55:42
2013-12-18T23:29:32	2013-12-18	23:29:32

Affects 1 column, all rows

Creates 2 columns

Extract on: '{upper}'

DATETIME	DATETIME1
2013-12-19T8:49:19	T
2013-12-30T1:55:42	T
2013-12-18T23:29:32	T

Affects 1 column, all rows

Creates 1 column

DURATION	#	DISCONNECT_REASON	ABC	MSWICENT	ABC	BASCENTCONT
	11 - 19			1 Category		2 Categories
	11			MSC001		BSC001
	11			MSC001		BSC001
	11			MSC001		BSC002
	11			MSC001		BSC002
	11			MSC001		BSC002
	11			MSC001		BSC001
	11			MSC001		BSC002
	11			MSC001		BSC001
	11			MSC001		BSC001
	11			MSC001		BSC001
	11			MSC001		BSC001

```

splitrows col: column1 on: '\n'
split col: column1 on: '\t' limit: 3
split col: column2 on: '\v' limit: 2
split col: column4 on: ':' limit: 2
replace col: column7 with: '' on: '>' global: true
replace col: column1 with: '' on: '<' global: true
split col: column3 on: '\v' limit: 2
header
split col: DATETIME on: '{upper}'

```

Split on: '{upper}'

DATETIME	DATETIME1	DATETIME2
2013-12-19T8:49:19	2013-12-19	8:49:19
2013-12-30T1:55:42	2013-12-30	1:55:42
2013-12-18T23:29:32	2013-12-18	23:29:32

Affects 1 column, all rows

Creates 2 columns

Extract on: '{upper}'

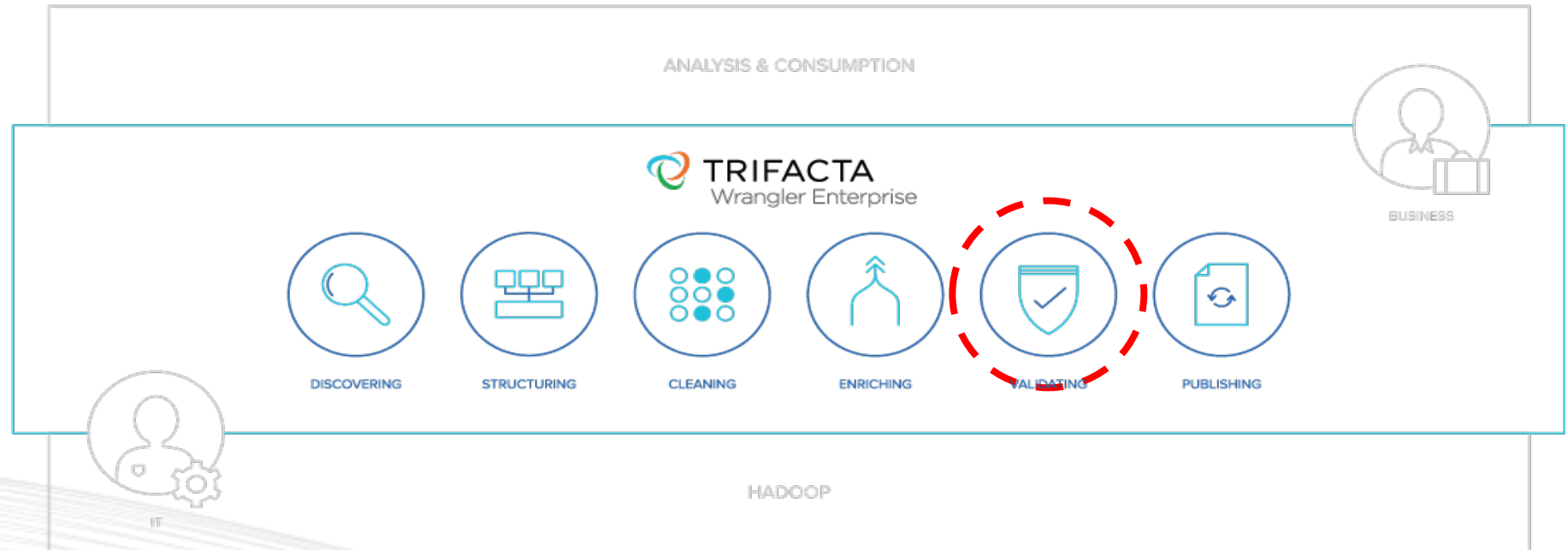
DATETIME	DATETIME1
2013-12-19T8:49:19	T
2013-12-30T1:55:42	T
2013-12-18T23:29:32	T

Affects 1 column, all rows

Creates 1 column

Where Profiling Fits In

We validate preparation via *profiling*

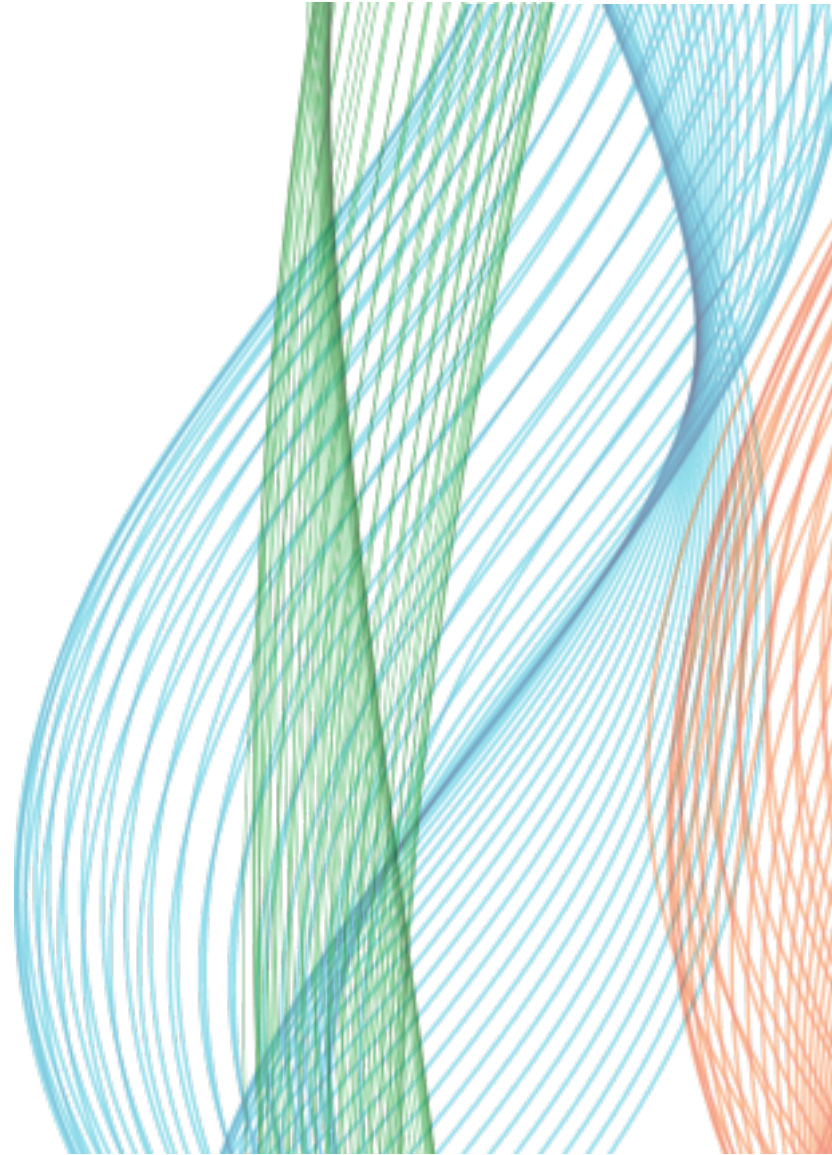


Spark Profiling at Trifacta

- Profiling results of transformation at scale
 - Validation through profiles
- Challenges
 - Scale
 - Automatic job generation
- Our solution
 - Spark profiling job server
 - JSON spec
 - Pay-as-you-go



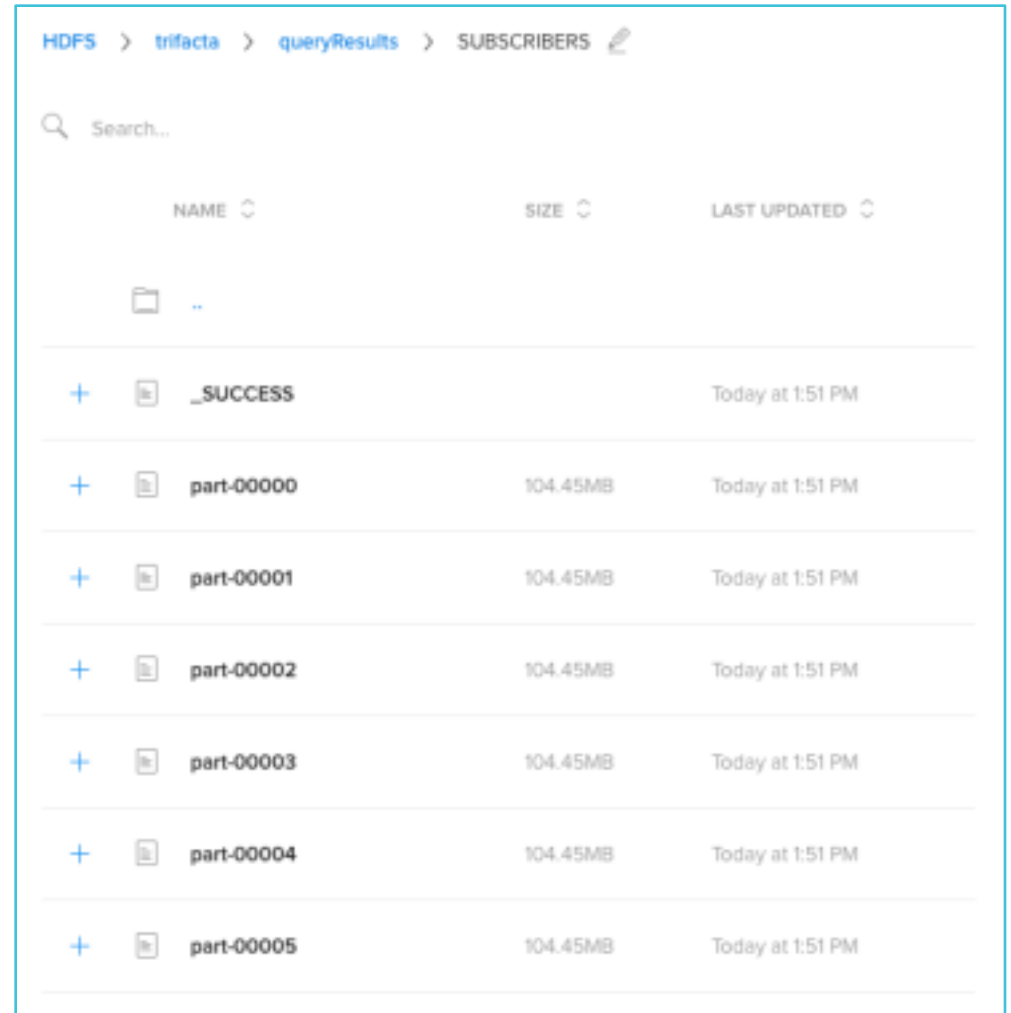
The Case for Profiling





Job Results

- Even clean raw data is not informative.
 - Especially when it is too large to inspect manually.
- Need a summary representation.
 - Statistical and visual
- Generate profile programmatically.



HDFS > trifacta > queryResults > SUBSCRIBERS		
Search...		
NAME	SIZE	LAST UPDATED
..		
+ [icon] _SUCCESS		Today at 1:51 PM
+ [icon] part-00000	104.45MB	Today at 1:51 PM
+ [icon] part-00001	104.45MB	Today at 1:51 PM
+ [icon] part-00002	104.45MB	Today at 1:51 PM
+ [icon] part-00003	104.45MB	Today at 1:51 PM
+ [icon] part-00004	104.45MB	Today at 1:51 PM
+ [icon] part-00005	104.45MB	Today at 1:51 PM



Visual Result Validation

The screenshot displays a web interface for a dataset named 'SUBSCRIBERS'. At the top, it shows 'Last updated: Today at 1:28 PM' and 'Created: Today at 1:28 PM'. Below this is a 'Jobs' section with four tabs: 'All', 'Complete', 'Failed', and 'Running'. The 'All' tab is selected. A job card is visible, showing 'JOB ID: 6', '1 Datasource', and 'Completed 01:33pm Jun 6'. The status 'Transform Job finished.' is displayed in green. At the bottom of the job card are icons for download and print.

- Similar to the profiling we saw above the data grid.
- Applied at scale, to the full result.
- Reveals mismatched, missing values.



Visual Result Validation

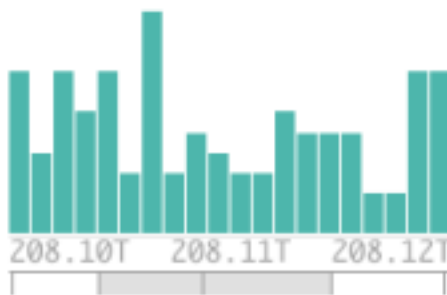
The screenshot displays a web interface for managing data jobs. At the top, a header bar shows 'SUBSCRIBERS' with a table icon, 'Last updated: Today at 1:28 PM', and 'Created: Today at 1:28 PM'. Below this is a 'Jobs' section with four tabs: 'All', 'Complete', 'Failed', and 'Running'. Two job cards are visible. The first card, for 'JOB ID: 6', shows '1 Datasource', 'Completed 01:33pm Jun 6', and the message 'Transform Job finished.' with download and print icons. The second card, for 'JOB ID: 5', shows '1 Datasource', 'Completed 01:32pm Jun 6', a progress bar, and a summary: '91% Valid', '3% Mismatched', and '7% Missing'. It includes a 'View Results' button and download/print icons.

Job ID	Status	Completion Time	Summary
JOB ID: 6	Completed	01:33pm Jun 6	Transform Job finished.
JOB ID: 5	Completed	01:32pm Jun 6	91% Valid, 3% Mismatched, 7% Missing

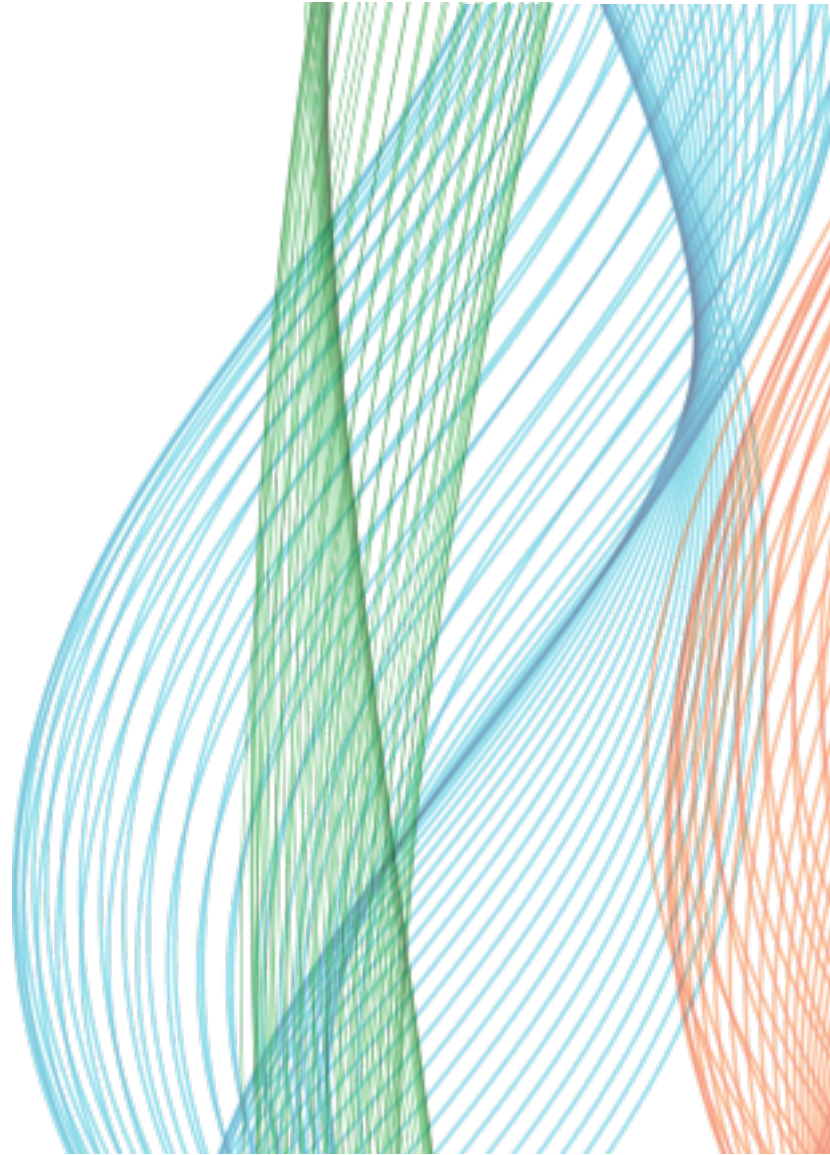
- Similar to the profiling we saw above the data grid.
- Applied at scale, to the full result.
- Reveals mismatched, missing values.



Visual Result Validation

#	IMSI	⌚	CONTRACT_END	@	EMAIL	ABC	OCCUPATION
Valid	3,531	Valid	3,531	Valid	3,009	Valid	1,648
Mismatched	0	Mismatched	0	Mismatched	522	Mismatched	0
Empty	107	Empty	107	Empty	107	Empty	1,990
				Top 20 values		Top 7 values	
Minimum 208,100,112,262,...		Minimum Nov 17 2009		ipsum.leo@felis.co.uk 10		Unemployed 380	
Lower quartile 208,104,019,2...		Lower quartile Feb 19 2014		montes@orci.co.uk 10		Director 229	
Median 208,108,656,229,74...		Median Oct 23 2014		cubilia.Curae.Donec@N... 10		Software Developer 229	
Upper quartile 208,114,546,6...		Upper quartile Apr 28 2015		eros.non.enim@Curabit... 10		Employee 227	
Maximum 208,119,614,632,...		Maximum Dec 29 2015		enim@magna.com 10		Salesman 227	
				posuere.vulputate.lac... 10		Architect 203	
				facilisis.facilisis@i... 9		Administrator 153	
				rutrum@tellusfaucibus... 9			
				Vivamus@quisdiam.net 9			
				biglia@gmail.com 9			
				fringilla@condimentum... 9			
				non@sitametrisus.edu 9			
				condimentum@Morbisita... 9			
				In@Donecnibhenim.net 9			
				eu@nonfeugiat.net 9			
				eu@nequeseddictum.com 9			
				sociis.natoque@ipsuml... 9			
				varius@erat.com 9			
				purus.sapien@adipisci... 9			
				adipiscing@lacusUtne... 9			

Challenges



The Goal: Profiling for Every Job

- We've talked about the value of profiling, but...
- Profiling is not a tablestakes feature, at least not on day 1.
 - Don't want our users to disable it!
- Profiling is potentially more expensive than transformation.



Profiling at Scale

- Large data volume
 - Long running times
 - Memory constraints



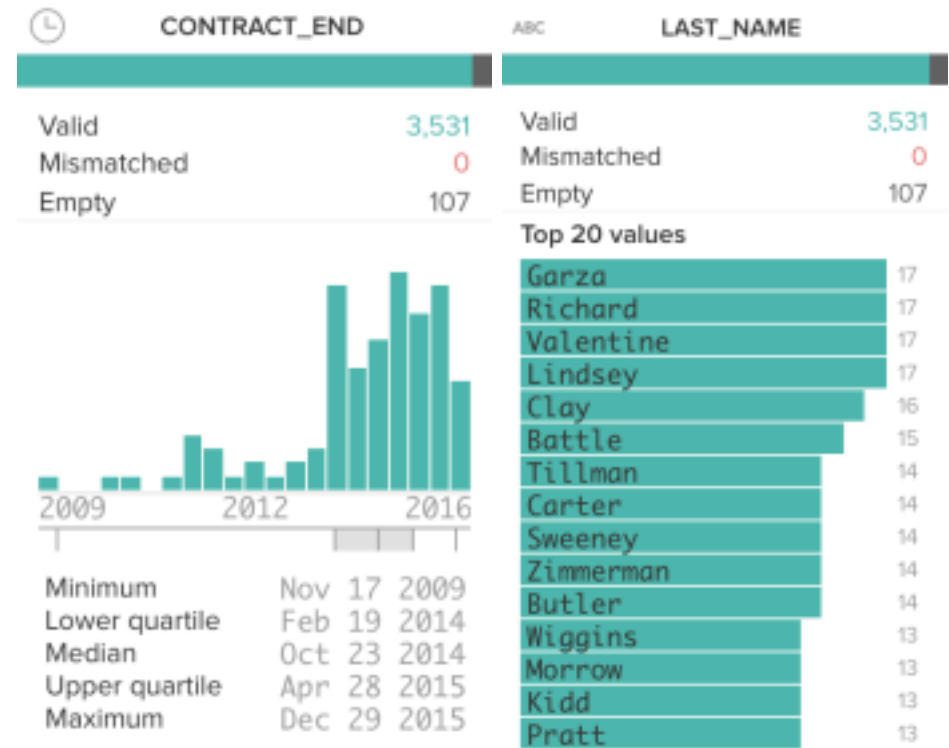
Performance vs. Accuracy

- Approximation brings performance gains while still meeting profiler summarization requirements.
- Off-the-shelf libraries (*count-min-sketch*, *T-digest*) great for approximating counts and non-distributive stats.
- Not a silver bullet though...sometimes approximations are confusing.



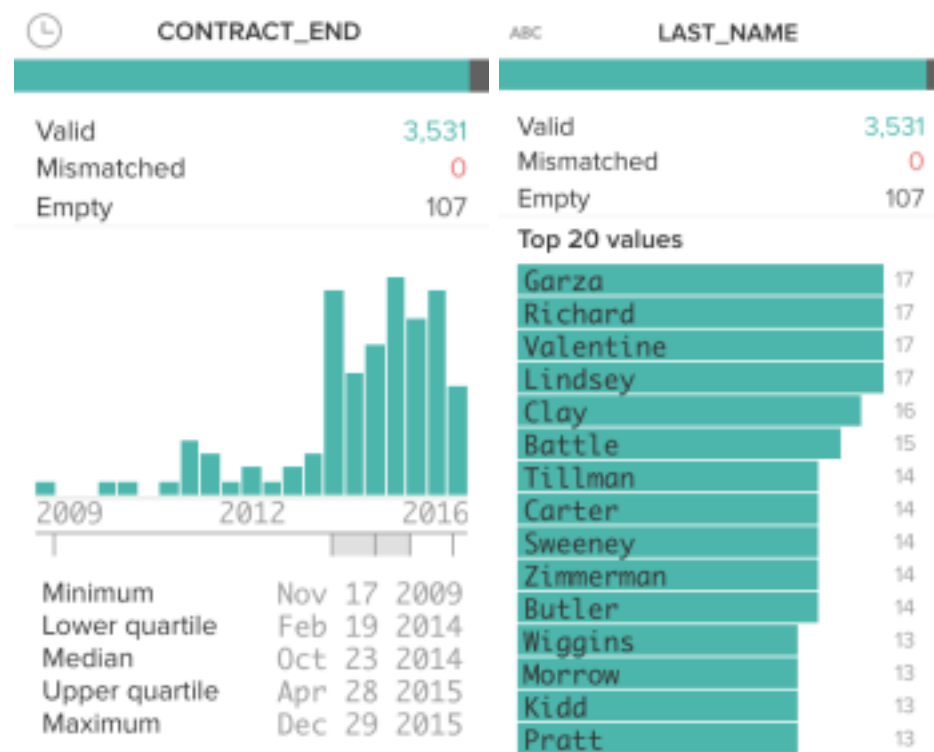
Flexible Job Generation

- Profile for all users, all use cases: **not** a one-off
 - Diverse schemas
 - Any number of columns
- Calculate statistics for all data types
 - Numeric vs. categorical
 - Container types (maps, arrays,...)
 - User-defined



Flexible Job Generation

- Profile for all users, all use cases: **not** a one-off
 - Diverse schemas
 - Any number of columns
- Calculate statistics for all data types
 - Numeric vs. categorical
 - Container types (maps, arrays,...)
 - User-defined



Transformation vs. Profiling

- Transformation is primarily row-based.
- Profiling is column-based.
- Different execution concerns.



Source		to be dropped		Preview			
	DATETIME2	#	DATETIME3	#	DATETIME4	#	DATETIME5
							
00:00 - 23:00			0 - 25		0 - 59		0 - 59
8:49:19		8		49		19	
1:55:42		1		55		42	
23:29:32		23		29		32	
19:52:25		19		52		25	
10:02:48		10		02		48	
7:50:56		7		50		56	
5:57:43		5		57		43	
22:42:20		22		42		20	

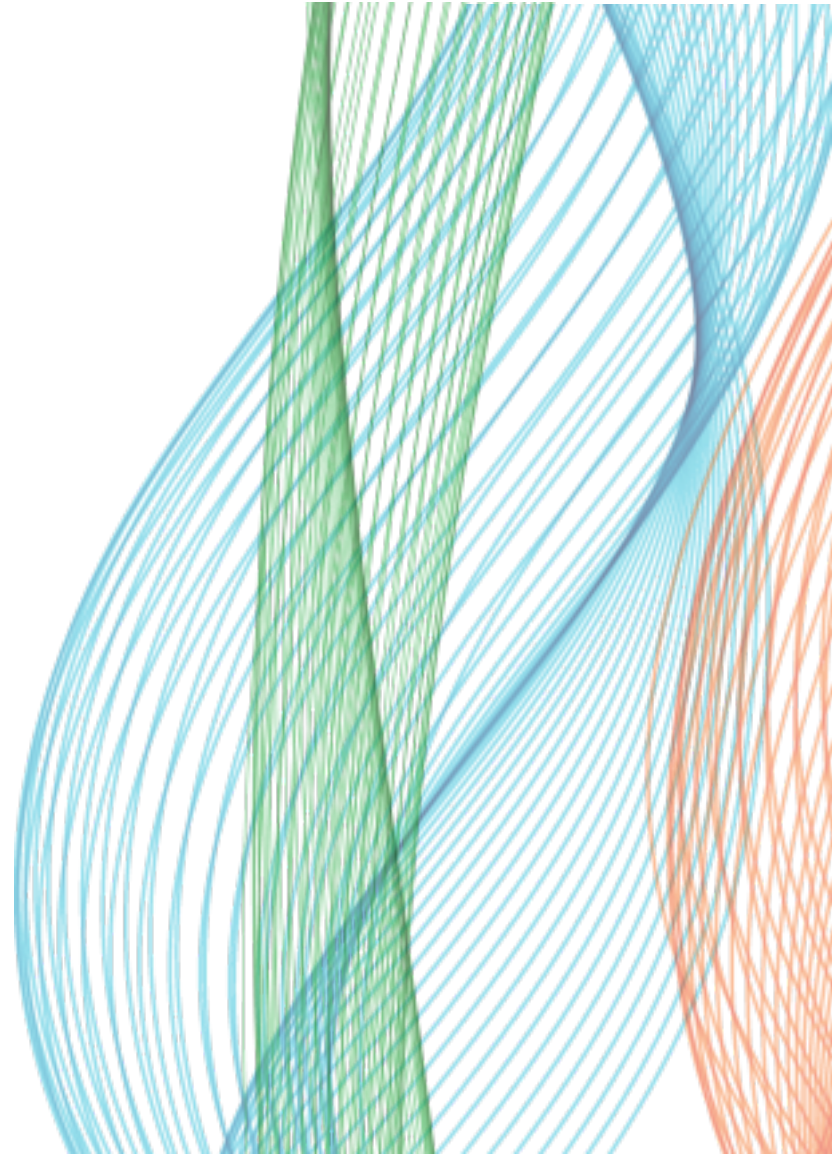
Value		Group By	
#	DURATION	#	DISCONNECT_REASON
0 - 451		11 - 19	
52		11	
327		11	
284		11	
326		11	
281		11	
302		11	
328		11	
328		11	
274		11	
290		11	
307		11	
303		11	
292		11	
315		11	



Preview			
#	DISCONNECT_REASON	#	sum_DURATION
11 - 19	0 - 303.72k		
11	303724		
19	24321		
	0		



Solution: Spark in the Trifacta System



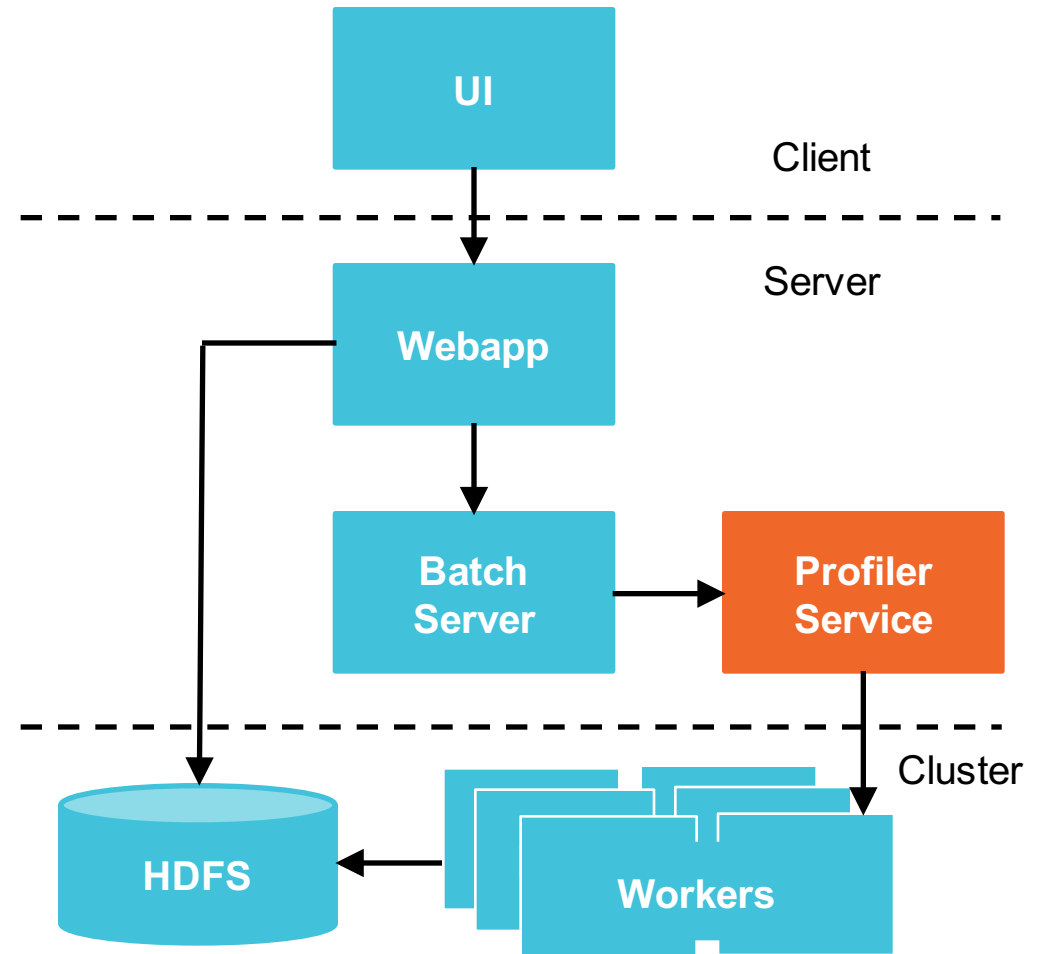
Why Spark

- Effective at OLAP
- Flexible enough for UDFs
- Not tied to distributions
- Mass adoption
- Low latency

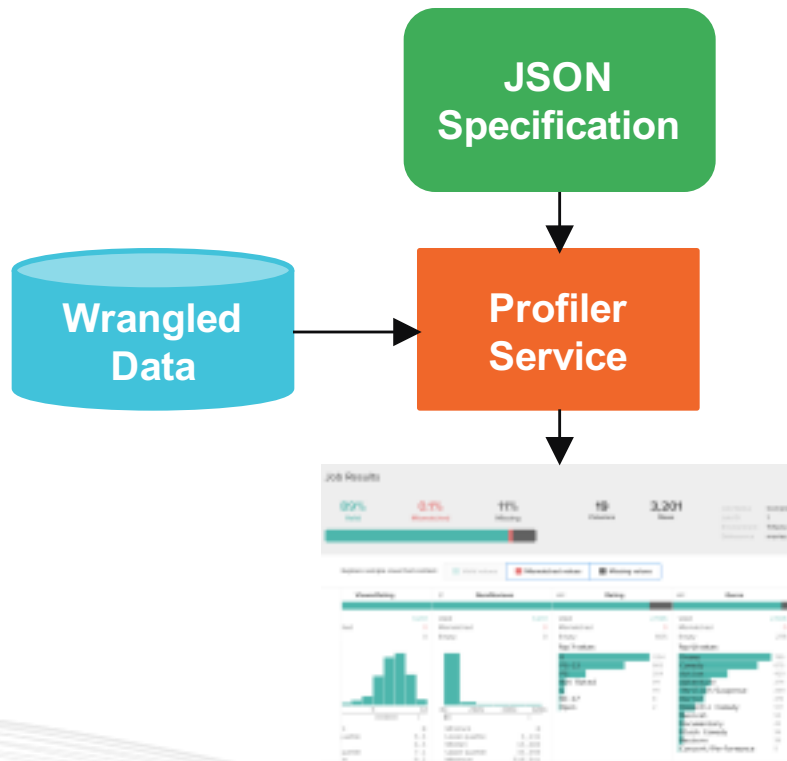


Spark Profile Jobs Server

- We built a Spark job server
- Automatically profiles output of transform jobs
- Outputs profiling results to HDFS
- Renders graphs in Trifacta product



Profiler Service



- Which columns to profile and their types
- Uses Trifacta type system, eg:
 - Dates
 - Geography
 - User-defined
- Requested profiling statistics
 - Histograms
 - Outliers
 - Empty, invalid, valid
- Extensible
 - Pairwise statistics
 - User-defined functions



Profiling DSL

```
{
  "input": "hdfs://hadoop.trifacta-dev.net:8020:/trifacta...",
  "schema": {
    "order": ["column1", "column2", "column3"],
    "types": {
      "column1": ["Datetime", {"regexes": , "groupLocs": {...}}],
      "column2": [...],
      "column3": [...]
    }
  },
  "commands": [
    {
      "column": "*",
      "output": "hdfs://hadoop.trifacta-dev.net:8020:/trifacta...",
      "profiler-type": "histogram",
      "params": {...}
    },
    {
      "column": "column1",
      "output": "hdfs://hadoop.trifacta-dev.net:8020:/trifacta...",
      "profiler-type": "type-check",
      "params": {...}
    }
  ]
}
```



Profiling DSL

```
{
  "input": "hdfs://hadoop.trifacta-dev.net:8020:/trifacta...",
  "schema": {
    "order": ["column1", "column2", "column3"],
    "types": {
      "column1": ["Datetime", {"regexes": ..., "groupLocs": {...}}],
      "column2": [...],
      "column3": [...]
    }
  },
  "commands": [
    {
      "column": "*",
      "output": "hdfs://hadoop.trifacta-dev.net:8020:/trifacta...",
      "profiler-type": "histogram",
      "params": {...}
    },
    {
      "column": "column1",
      "output": "hdfs://hadoop.trifacta-dev.net:8020:/trifacta...",
      "profiler-type": "type-check",
      "params": {...}
    }
  ]
}
```



Profiling DSL

```
{
  "input": "hdfs://hadoop.trifacta-dev.net:8020:/trifacta...",
  "schema": {
    "order": ["column1", "column2", "column3"],
    "types": {
      "column1": ["Datetime", {"regexes": , "groupLocs": {...}}],
      "column2": [...],
      "column3": [...]
    }
  },
  "commands": [
    {
      "column": "*",
      "output": "hdfs://hadoop.trifacta-dev.net:8020:/trifacta...",
      "profiler-type": "histogram",
      "params": {...}
    },
    {
      "column": "column1",
      "output": "hdfs://hadoop.trifacta-dev.net:8020:/trifacta...",
      "profiler-type": "type-check",
      "params": {...}
    }
  ]
}
```



Profiling DSL

```
{
  "input": "hdfs://hadoop.trifacta-dev.net:8020:/trifacta...",
  "schema": {
    "order": ["column1", "column2", "column3"],
    "types": {
      "column1": ["Datetime", {"regexes": , "groupLocs": {...}}],
      "column2": [...],
      "column3": [...]
    }
  },
  "commands": [
    {
      "column": "*",
      "output": "hdfs://hadoop.trifacta-dev.net:8020:/trifacta...",
      "profiler-type": "histogram",
      "params": {...}
    },
    {
      "column": "column1",
      "output": "hdfs://hadoop.trifacta-dev.net:8020:/trifacta...",
      "profiler-type": "type-check",
      "params": {...}
    }
  ]
}
```



Performance Improvements

Spark profiling speed-up vs. MapReduce

	Few Columns	Many Columns
High Cardinality	2X	4X
Low Cardinality	10X	20X

- A few troublesome use cases:
 - High cardinality
 - Non-distributive stats
- Huge gains for large numbers of columns



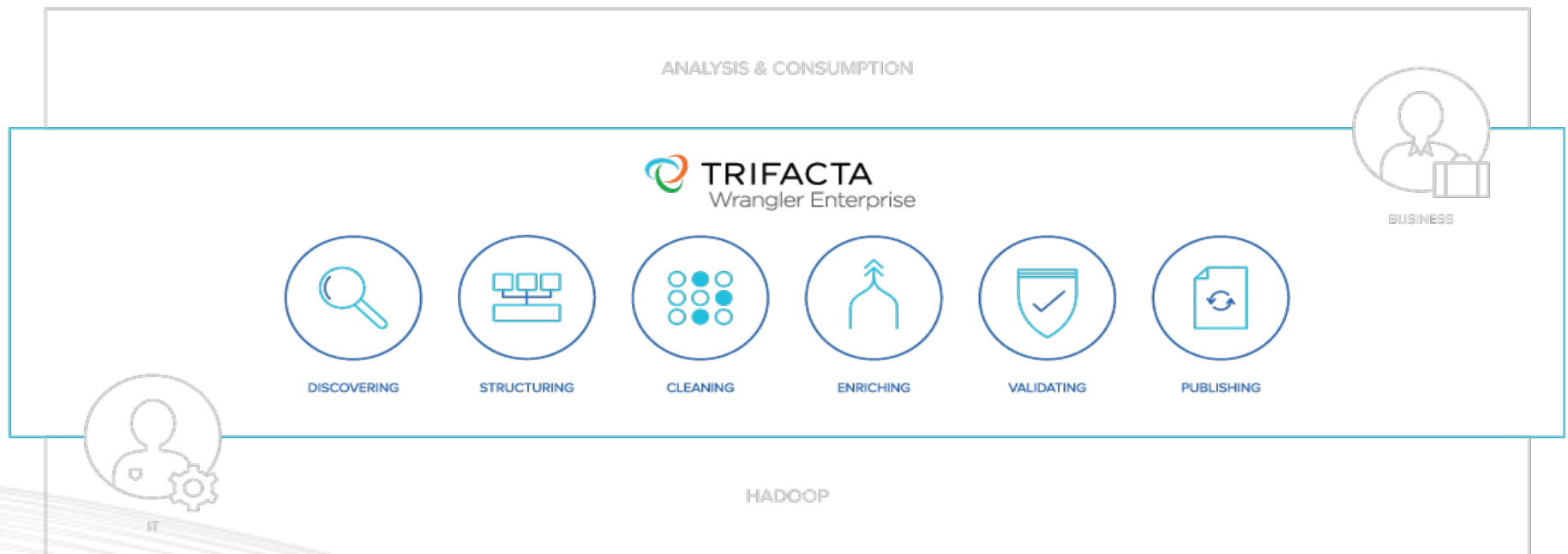
Pay-as-you-go Profiling

- Users do not always need profiles for all columns
- More complex and expensive statistics
- Drilldown
- Iterative exploration



Conclusion

Profiling informs data preparation.



Questions?

