Wrangler: Interactive Visual Specification of Data Transformation Scripts

Problem Statement

- Big data: huge amounts of unstructured data from plethora of sources
- Data must be structured to make it palatable for databases, statistical packages, and visualization tools
- Issues to be addressed: misspellings, missing data, unresolved duplicates, outliers..
- According to an estimate: Data Cleaning accounts for 80% of the development time and cost in Data Warehousing projects!

Traditional Data Wrangling

- Writing idiosyncratic scripts in programming languages like Python, Perl etc.

- Manual editing in Microsoft Excel

- Highly tedious processes and could easily discourage one from working with data

- But we cannot!

Because in data analysis practice, useful messages lie in these tedious processes

Also…

- In the overall data lifecycle, transforming and cleaning the data constitutes only the first step
- Data updates and evolving schemas necessitate the reuse of data transformations
- Analysts who use the transformed data might wish to reuse and refine the transformations previously applied
- As a result, the proper output of data wrangling constitutes two main aspects:
  - the transformed data
  - an editable and auditable description of the applied transformations

- Hence, Wrangler! A system for interactive data transformation

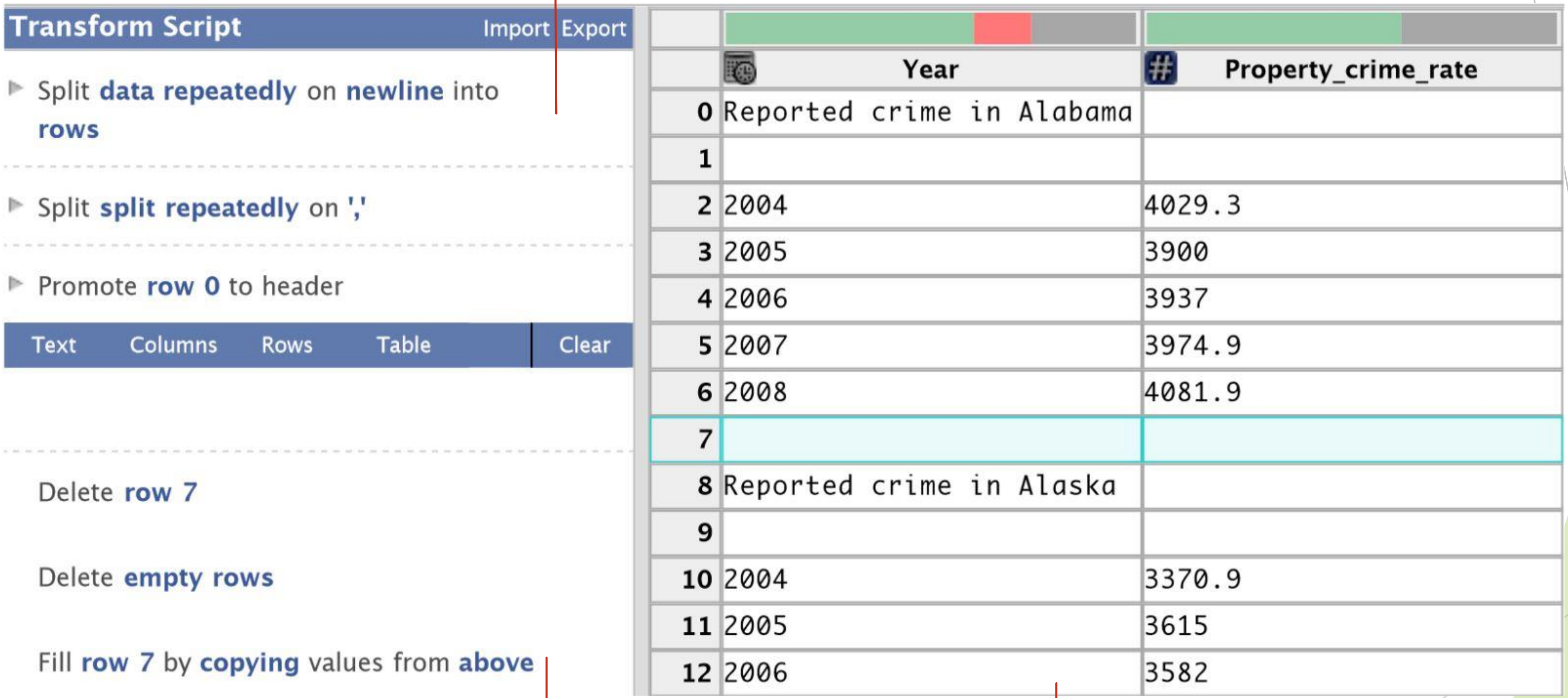
Wrangler

- Couples a *mixed-initiative user interface* and a *declarative transformation language*
- Transformations on data are build by a sequence of basic transforms
- When a user selects data,
  - Wrangler suggests a sequence of transforms that can be applied in that context
  - Transform suggestions are provided in *natural language descriptions* with *interactive parameters*
  - *Visual previews* of transforms are provided
  - An *interactive history viewer* is maintained
  - Wrangler scripts can be run in a web browser using **JavaScript** or can be translated to MapReduce or Python code

Example

Fig 1: The Wrangler Interface

History of transforms



Transform Script

- Split **data repeatedly** on **newline** into **rows**
- Split **split repeatedly** on **','**
- Promote **row 0** to header

Delete **row 7**

Delete **empty rows**

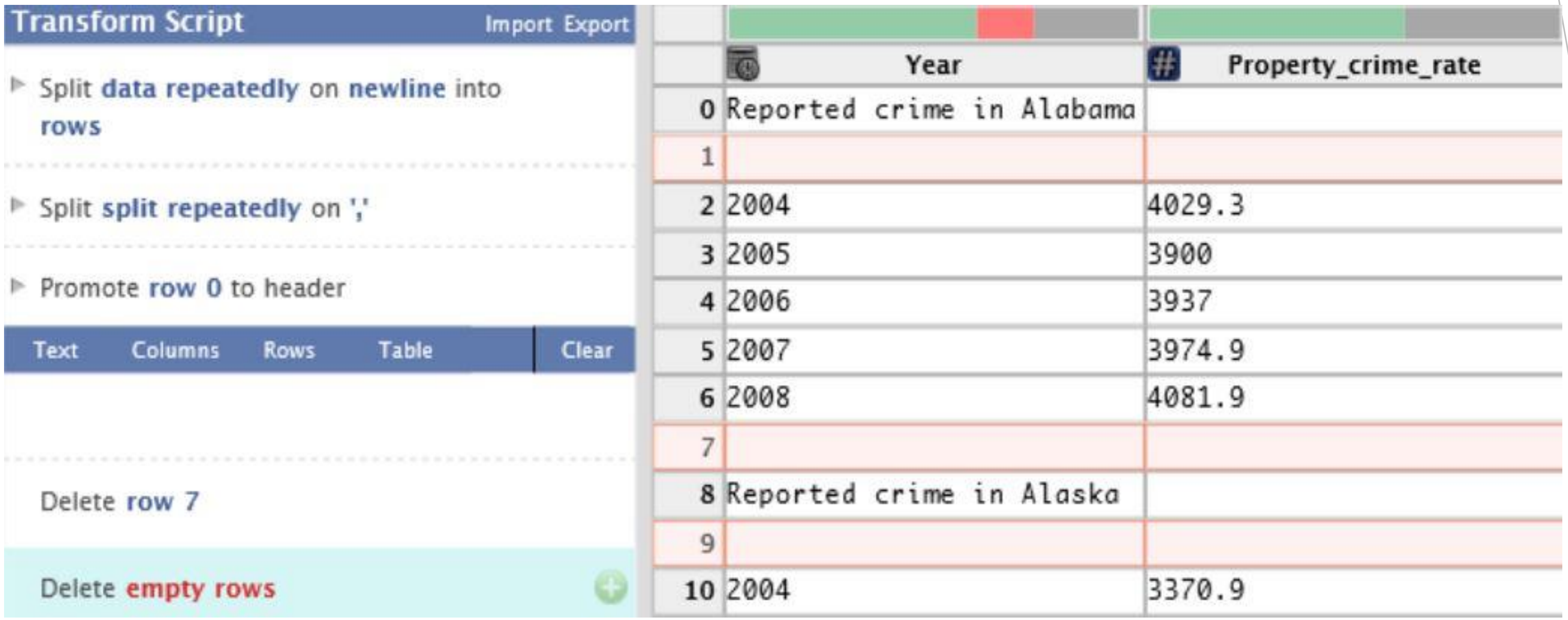
Fill **row 7** by **copying** values from **above**

	Year	Property_crime_rate
0	Reported crime in Alabama	
1		
2	2004	4029.3
3	2005	3900
4	2006	3937
5	2007	3974.9
6	2008	4081.9
7		
8	Reported crime in Alaska	
9		
10	2004	3370.9
11	2005	3615
12	2006	3582

Transform selection menu

Interactive data table

…continued



- Split data repeatedly on newline into rows
- Split split repeatedly on ','
- Promote row 0 to header

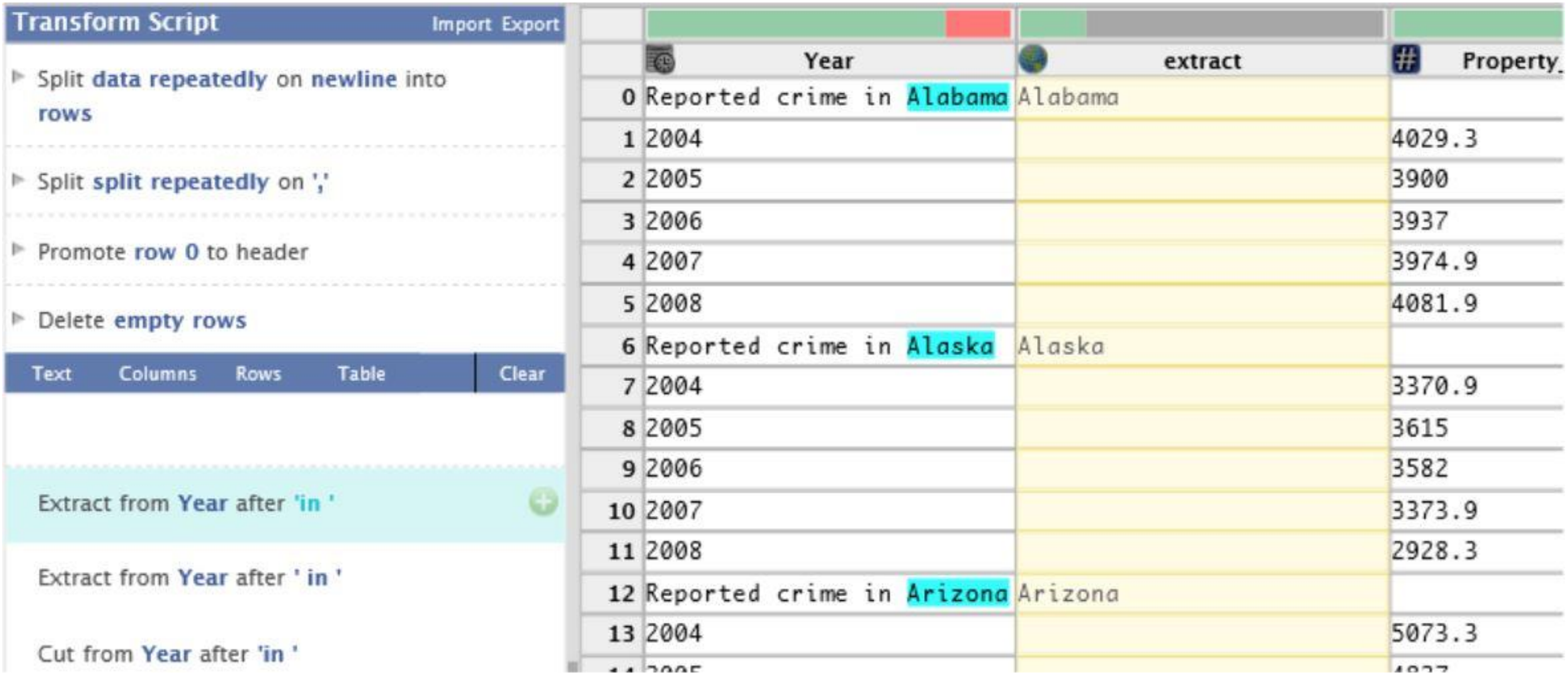
Delete row 7

Delete empty rows

	Year	Property_crime_rate
0	Reported crime in Alabama	
1		
2	2004	4029.3
3	2005	3900
4	2006	3937
5	2007	3974.9
6	2008	4081.9
7		
8	Reported crime in Alaska	
9		
10	2004	3370.9

Fig 2: Deletion of empty rows

…continued



- Split data repeatedly on newline into rows
- Split split repeatedly on ','
- Promote row 0 to header
- Delete empty rows

Extract from Year after 'in '

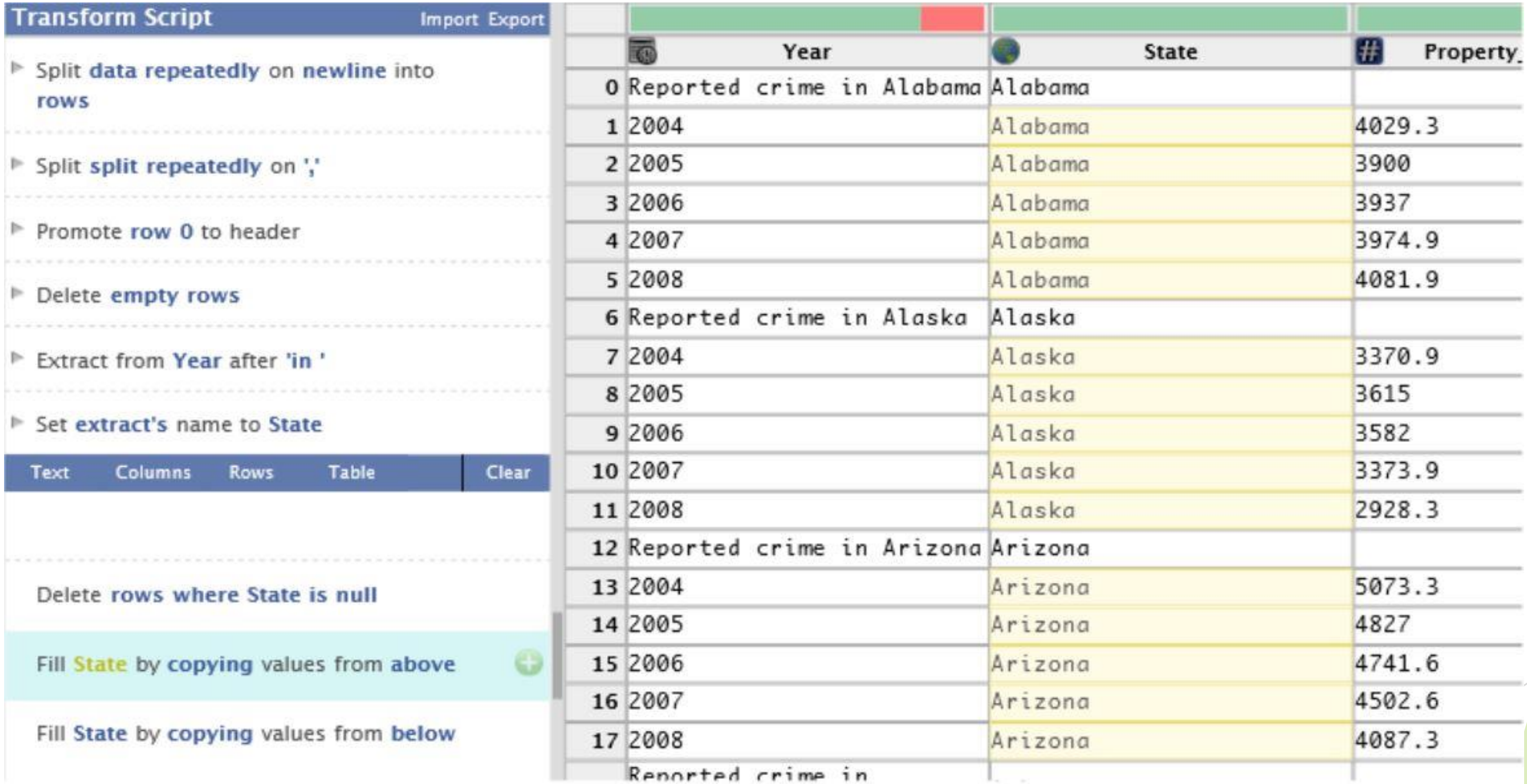
Extract from Year after ' in '

Cut from Year after 'in '

	Year	extract	Property_
0	Reported crime in Alabama	Alabama	
1	2004		4029.3
2	2005		3900
3	2006		3937
4	2007		3974.9
5	2008		4081.9
6	Reported crime in Alaska	Alaska	
7	2004		3370.9
8	2005		3615
9	2006		3582
10	2007		3373.9
11	2008		2928.3
12	Reported crime in Arizona	Arizona	
13	2004		5073.3

Fig 3: Extracting state names

…continued



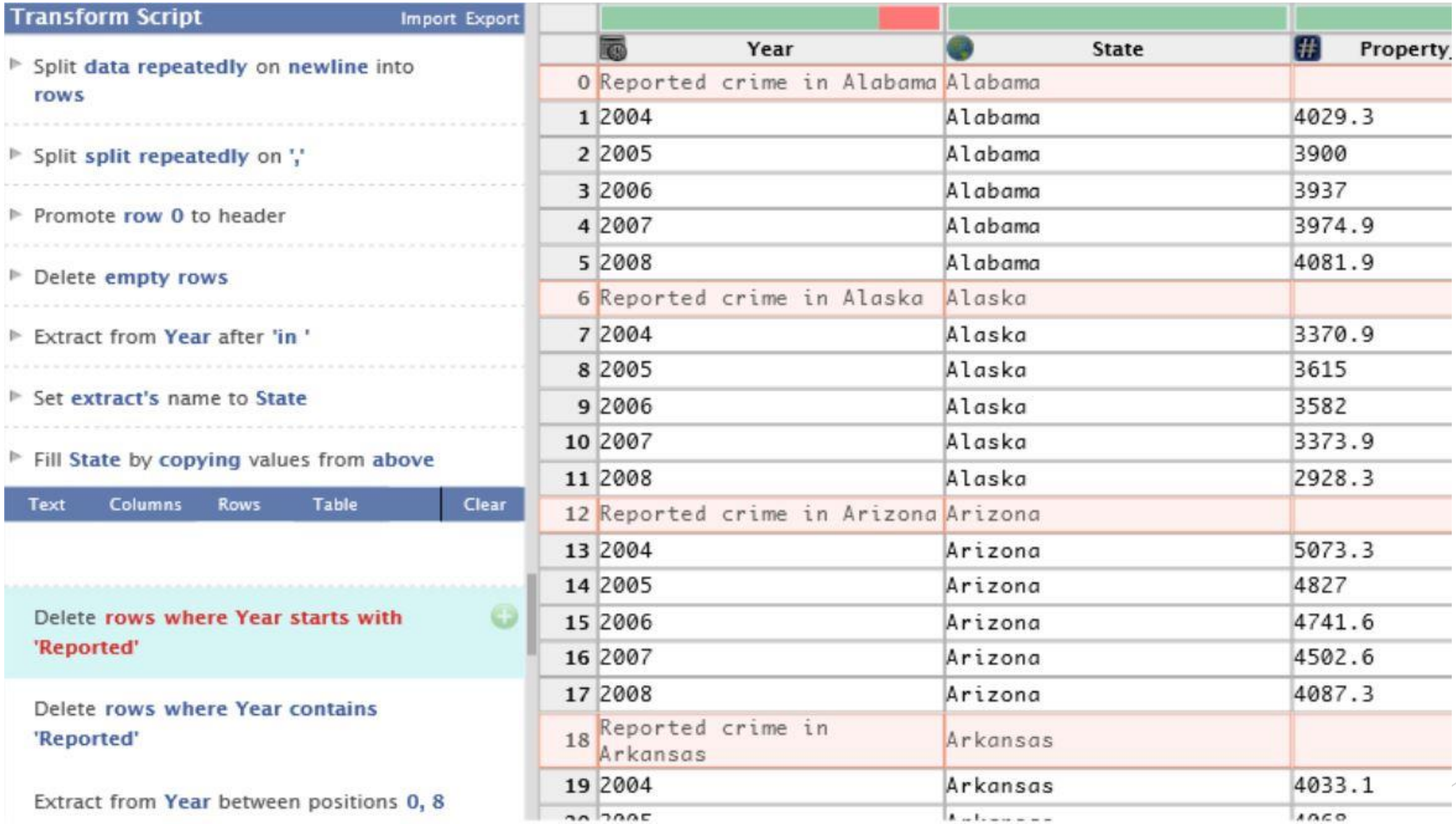
- Split data repeatedly on newline into rows
- Split split repeatedly on ','
- Promote row 0 to header
- Delete empty rows
- Extract from Year after 'in '
- Set extract's name to State

- Delete rows where State is null
- Fill State by copying values from above
- Fill State by copying values from below

	Year	State	Property_
0	Reported crime in Alabama	Alabama	
1	2004	Alabama	4029.3
2	2005	Alabama	3900
3	2006	Alabama	3937
4	2007	Alabama	3974.9
5	2008	Alabama	4081.9
6	Reported crime in Alaska	Alaska	
7	2004	Alaska	3370.9
8	2005	Alaska	3615
9	2006	Alaska	3582
10	2007	Alaska	3373.9
11	2008	Alaska	2928.3
12	Reported crime in Arizona	Arizona	
13	2004	Arizona	5073.3
14	2005	Arizona	4827
15	2006	Arizona	4741.6
16	2007	Arizona	4502.6
17	2008	Arizona	4087.3
	Reported crime in		

Fig 4: Filling in missing values by copying values from above

…continued



- Split data repeatedly on newline into rows
- Split split repeatedly on ','
- Promote row 0 to header
- Delete empty rows
- Extract from Year after 'in '
- Set extract's name to State
- Fill State by copying values from above

Delete rows where Year starts with 'Reported'

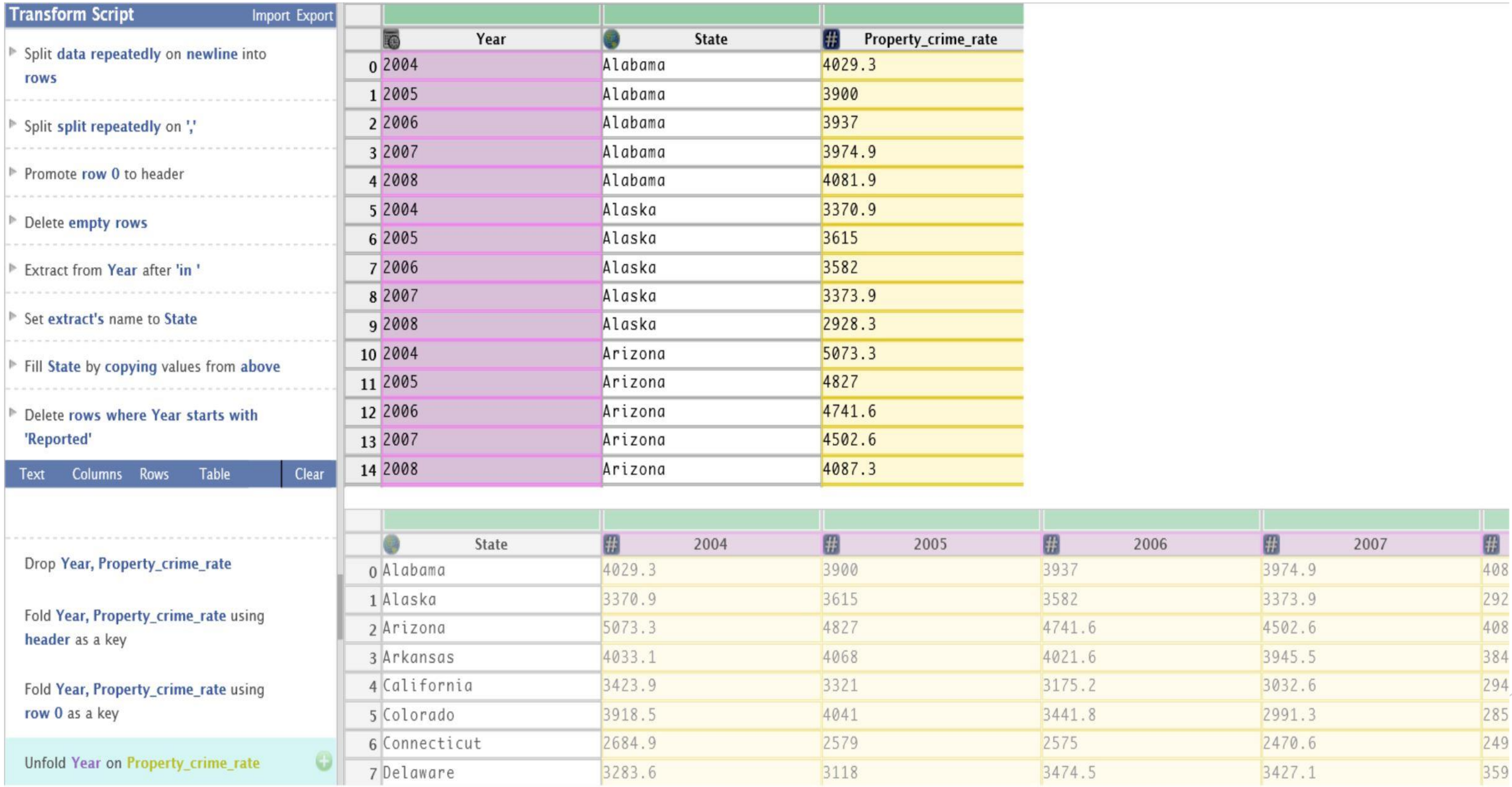
Delete rows where Year contains 'Reported'

Extract from Year between positions 0, 8

	Year	State	Property
0	Reported crime in Alabama	Alabama	
1	2004	Alabama	4029.3
2	2005	Alabama	3900
3	2006	Alabama	3937
4	2007	Alabama	3974.9
5	2008	Alabama	4081.9
6	Reported crime in Alaska	Alaska	
7	2004	Alaska	3370.9
8	2005	Alaska	3615
9	2006	Alaska	3582
10	2007	Alaska	3373.9
11	2008	Alaska	2928.3
12	Reported crime in Arizona	Arizona	
13	2004	Arizona	5073.3
14	2005	Arizona	4827
15	2006	Arizona	4741.6
16	2007	Arizona	4502.6
17	2008	Arizona	4087.3
18	Reported crime in Arkansas	Arkansas	
19	2004	Arkansas	4033.1

Fig 5: Type mismatch in column value detected; Wrangler suggests deletion

…continued



- Split data repeatedly on newline into rows
- Split split repeatedly on ','
- Promote row 0 to header
- Delete empty rows
- Extract from Year after 'in '
- Set extract's name to State
- Fill State by copying values from above
- Delete rows where Year starts with 'Reported'

Drop Year, Property_crime_rate

Fold Year, Property_crime_rate using header as a key

Fold Year, Property_crime_rate using row 0 as a key

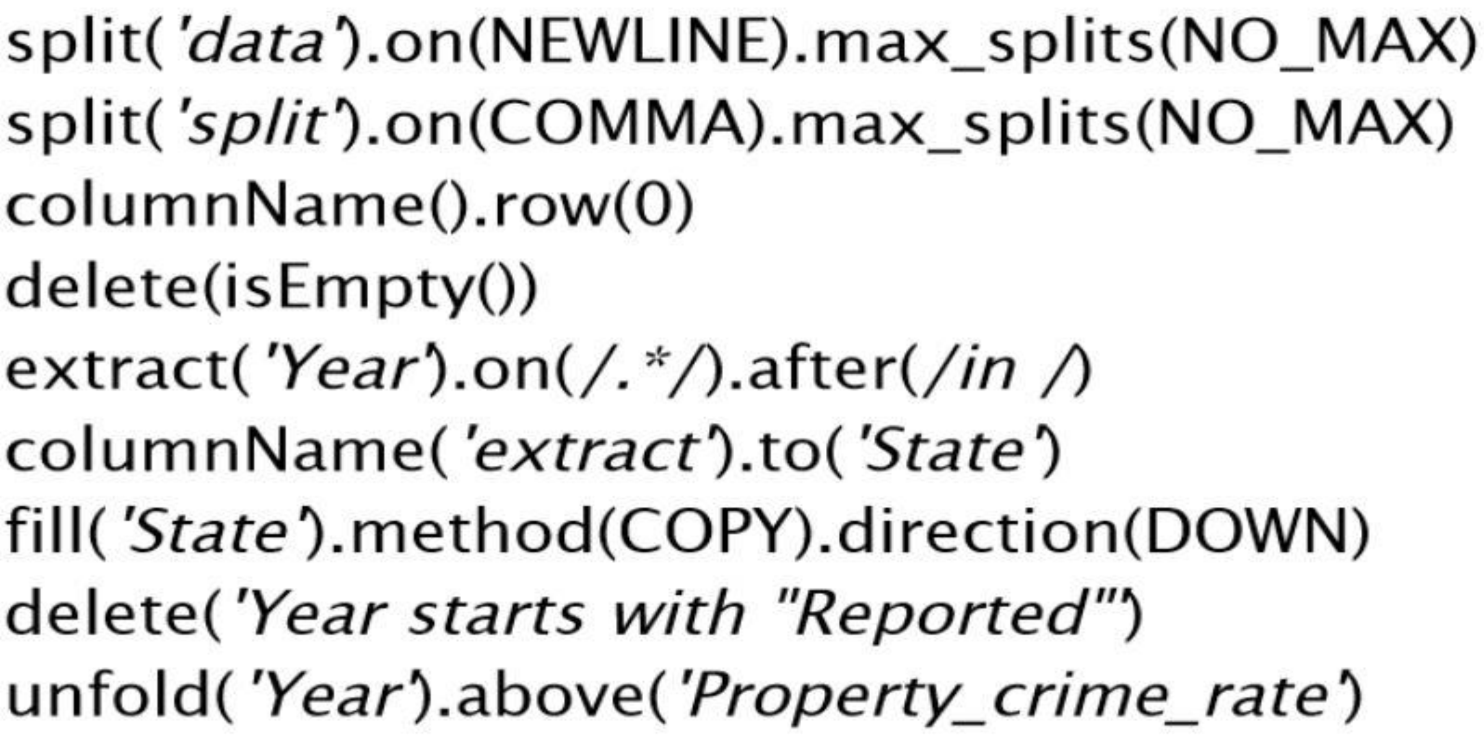
Unfold Year on Property_crime_rate

	Year	State	Property_crime_rate
0	2004	Alabama	4029.3
1	2005	Alabama	3900
2	2006	Alabama	3937
3	2007	Alabama	3974.9
4	2008	Alabama	4081.9
5	2004	Alaska	3370.9
6	2005	Alaska	3615
7	2006	Alaska	3582
8	2007	Alaska	3373.9
9	2008	Alaska	2928.3
10	2004	Arizona	5073.3
11	2005	Arizona	4827
12	2006	Arizona	4741.6
13	2007	Arizona	4502.6
14	2008	Arizona	4087.3

	State	2004	2005	2006	2007	
0	Alabama	4029.3	3900	3937	3974.9	408
1	Alaska	3370.9	3615	3582	3373.9	292
2	Arizona	5073.3	4827	4741.6	4502.6	408
3	Arkansas	4033.1	4068	4021.6	3945.5	384
4	California	3423.9	3321	3175.2	3032.6	294
5	Colorado	3918.5	4041	3441.8	2991.3	285
6	Connecticut	2684.9	2579	2575	2470.6	249
7	Delaware	3283.6	3118	3474.5	3427.1	359

Fig 6: Unfolding operation combining columns ‘Year’ and ‘Property_crime_rate’

Exporting generated script



```
split('data').on(NEWLINE).max_splits(NO_MAX)
split('split').on(COMMA).max_splits(NO_MAX)
columnName().row(0)
delete(isEmpty())
extract('Year').on(/.*/).after(/in /)
columnName('extract').to('State')
fill('State').method(COPY).direction(DOWN)
delete('Year starts with "Reported"')
unfold('Year').above('Property_crime_rate')
```

- The declarative data cleaning script, shown as JavaScript code
- A Wrangler runtime evaluates the script to produce transformed data

Wrangler Transformation language

- The Wrangler transformation language contains eight classes of transforms. These are:
  - **Map**
    - Map transforms map one input data row to zero, one, or multiple output rows
    - *Delete* transforms accept predicates determining which rows to remove
    - One-to-one transforms include splitting values into multiple columns
    - One-to-many transforms include splitting data into multiple rows
  - **Lookups and joins**
    - Incorporate data from external tables
    - Example, mapping zip codes to state names for aggregation across states
    - Wrangler currently supports equi-joins and approximate joins

…continued

- **Reshape Transforms**
  - Manipulate table structure and schema
  - Two reshaping operators provided – *fold* and *unfold*
  - *Fold* collapses multiple columns to two or more columns
  - *Unfold* creates new column headers from data values

- **Positional Transforms**
  - Include *fill* and *lag* operations
  - *Fill* operation generates values from neighbouring row/column values
  - *Lag* operation shifts the values of a column up/down by a specified number of rows

…continued

- The language also contains features for:
  - **Sorting, aggregation** (Ex. sum, min, max, mean, standard deviation)
  - **Key generation**
  - **Schema transforms** to set column names, specify column data types, and assign semantic roles
    - Wrangler supports standard data types (e.g., integers, numbers, strings)
    - Higher-level semantic roles (e.g., geographic location, classification codes, currencies)

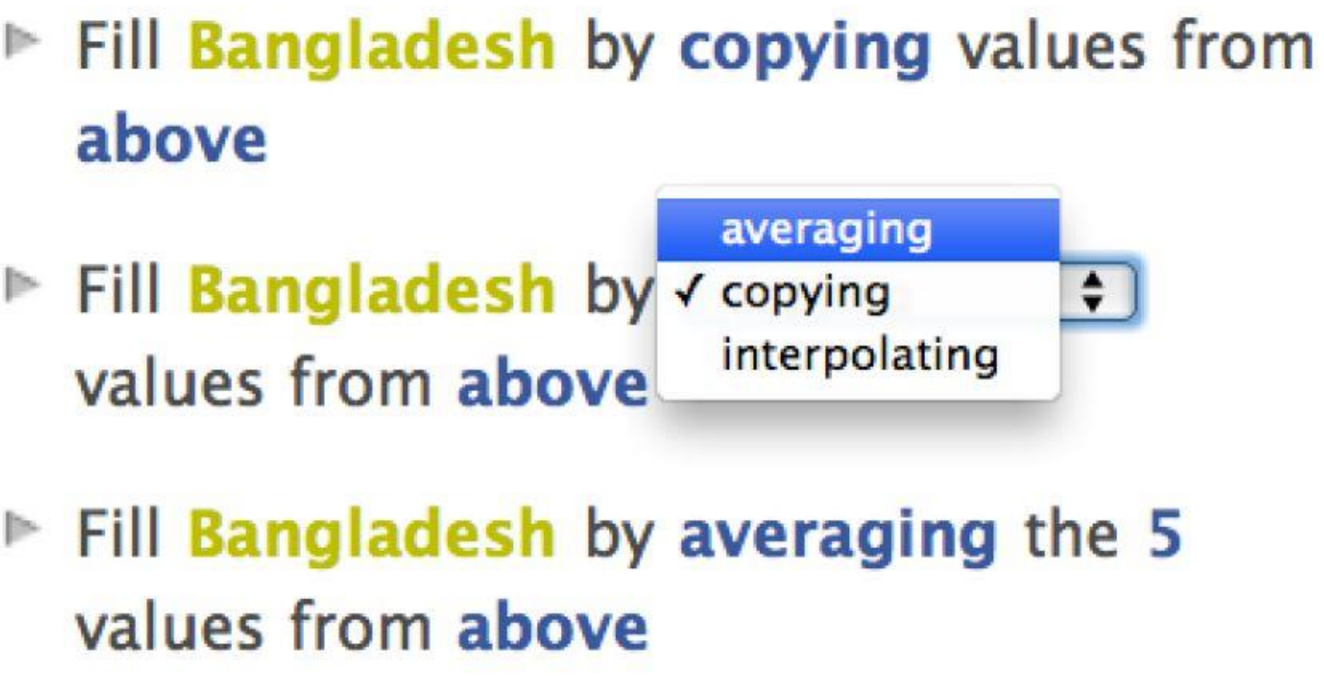
Wrangler Interface Design

- **Basic Interactions**
  - Supports six basic interactions within the data table
  - Users can – select rows, select columns, click bars in the data quality meter, select text within a cell, edit data values within the table, and assign column names, data types or semantic roles
  - Users can also choose transforms from the menu or refine suggestions by editing transform descriptions

- **Automated Transformation Suggestions**
  - As a user interacts with data, Wrangler generates a list of suggested transforms
  - The users can then,
    - provide more examples to disambiguate input to the inference engine
    - filter the space of transforms by selecting an operator from the transform menu
    - edit a transform by altering the parameters of a transform to a desired state

…continued

- **Natural Language Descriptions**
  - Wrangler generates short natural language descriptions of the transform type and parameters
  - These descriptions are editable, with parameters presented as bold hyperlinks



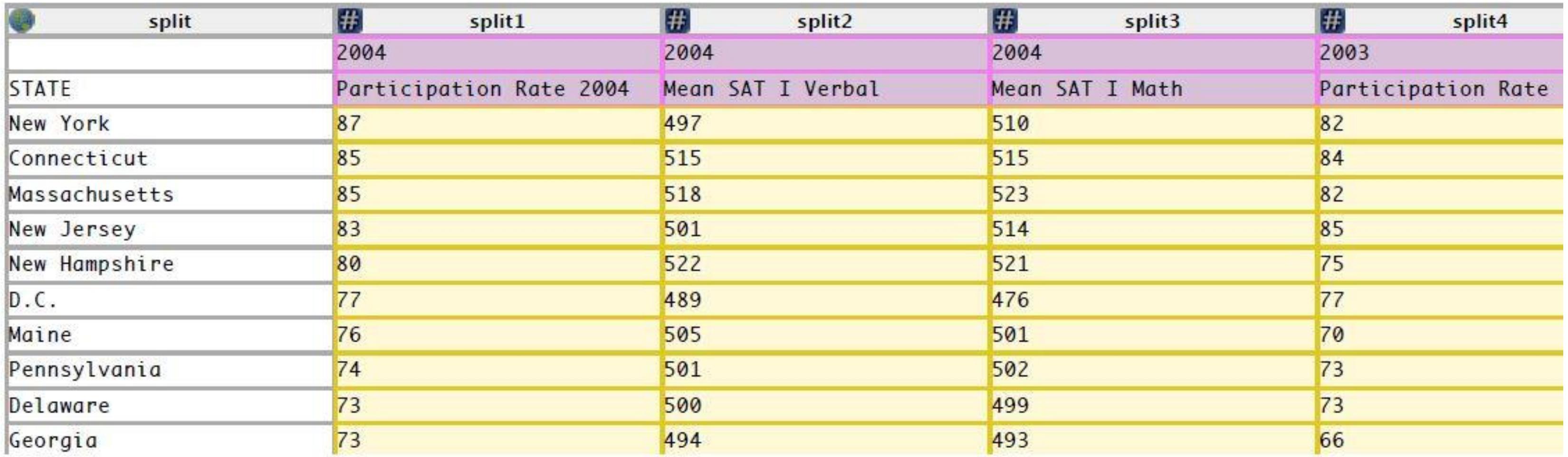
- Fill **Bangladesh** by **copying** values from **above**
- Fill **Bangladesh** by ✓ copying values from **above**
- Fill **Bangladesh** by **averaging** the **5** values from **above**

Editable Natural language Descriptions

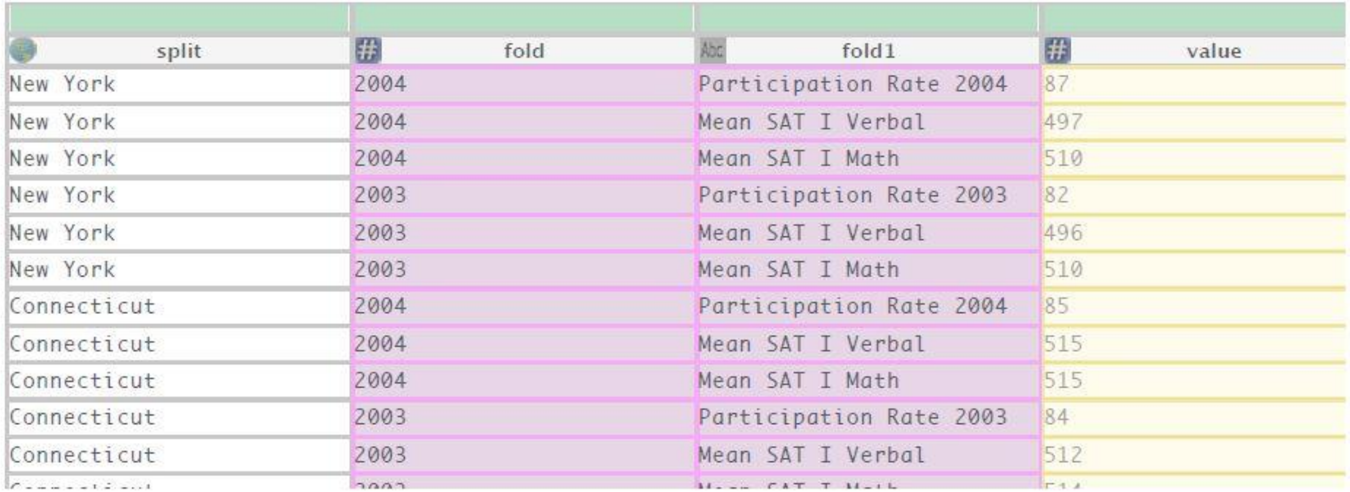
…continued

- **Visual Transformation Previews**
  - Wrangler uses visual previews to enable users to quickly evaluate the effect of a transform
  - Wrangler maps transforms to at least one of five preview classes: selection, deletion, update, column and table
    - Selection previews highlight relevant regions of text in all affected cells (Fig. 3)
    - Deletion previews color to-be-deleted cells in red (Fig. 2)
    - Update previews overwrite values in a column and indicate differences with yellow highlights (Fig. 4)
    - Column previews display new derived columns, e.g., as results from an extract operation (Fig. 3)
    - Fold and unfold transforms alter the structure of the table to such an extent that the best preview is to show another table (Fig. 6)

…continued



split	split1	split2	split3	split4
	2004	2004	2004	2003
STATE	Participation Rate 2004	Mean SAT I Verbal	Mean SAT I Math	Participation Rate
New York	87	497	510	82
Connecticut	85	515	515	84
Massachusetts	85	518	523	82
New Jersey	83	501	514	85
New Hampshire	80	522	521	75
D.C.	77	489	476	77
Maine	76	505	501	70
Pennsylvania	74	501	502	73
Delaware	73	500	499	73
Georgia	73	494	493	66



split	fold	fold1	value
New York	2004	Participation Rate 2004	87
New York	2004	Mean SAT I Verbal	497
New York	2004	Mean SAT I Math	510
New York	2003	Participation Rate 2003	82
New York	2003	Mean SAT I Verbal	496
New York	2003	Mean SAT I Math	510
Connecticut	2004	Participation Rate 2004	85
Connecticut	2004	Mean SAT I Verbal	515
Connecticut	2004	Mean SAT I Math	515
Connecticut	2003	Participation Rate 2003	84
Connecticut	2003	Mean SAT I Verbal	512
Connecticut	2003	Mean SAT I Math	514

Visual preview of a *fold* operation

…continued

- **Transformation Histories and Export**
  - As successive transforms are applied, Wrangler adds their descriptions to an interactive *transformation history viewer*
  - Wrangler then runs the generated script and updates the data table
  - Wrangler scripts also support lightweight text annotations. These annotations appear as comments in code-generated scripts
  - Users can export both generated scripts and transformed data. Analysts can later run saved or exported scripts on new data sources, modifying the script as needed

Wrangler Inference Engine

- Wrangler inference engine is responsible for generating a ranked list of suggested transforms
- Inputs to the engine consist of the following user interactions:
  - the current working transform
  - data descriptions such as column data types, semantic roles, and summary statistics
  - a corpus of historical usage statistics

- Transform suggestion proceeds in three phases:
  - inferring transform parameters from user interactions
  - generating candidate transforms from inferred parameters
  - ranking the results

…continued

- **Usage Corpus and Transform Equivalence**
  - To generate and rank transforms, Wrangler’s inference engine relies on a corpus of usage statistics
  - The corpus consists of frequency counts of transform descriptors and initiating interactions
  - In order to get useful transform frequencies, we define a relaxed matching routine. Two transforms are considered equivalent in our corpus if,
    - they have an identical transform type (e.g., extract or fold)
    - they have equivalent parameters. The four basic types of parameters are: row, column or text selections and enumerables

…continued

- **Inferring Parameter Sets from User Interaction**
  - In response to user interaction, Wrangler attempts to infer three types of transform parameters: row, column, or text selections
  - Each parameter's values are inferred independent of the other parameters. For example,
    - regular expressions for text selection are inferred based solely on the selected text
    - row selections are inferred based on row indices and predicate matching
    - for column selections, the columns that users have interacted with are returned

- **Generating Suggested Transforms**
  - After inferring parameter sets, Wrangler generates a list of transform suggestions
  - It instantiates each emitted transform with parameters from the parameter set
  - Wrangler then filters the suggestion set to remove "degenerate" transforms that would have no effect on the data

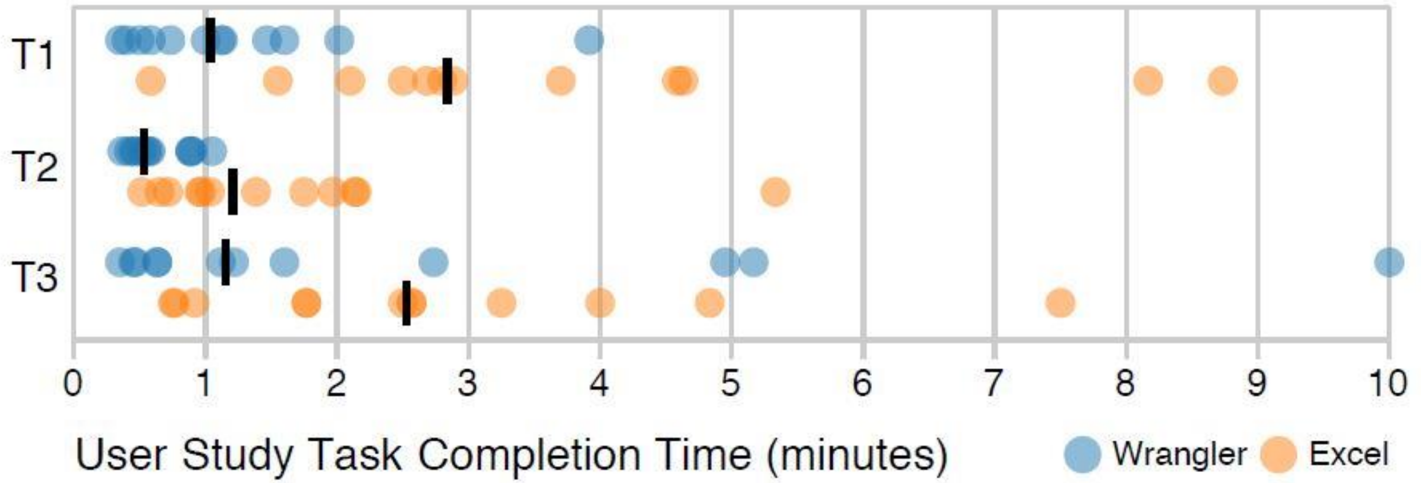
…continued

- **Ranking Suggested Transforms**
  - Wrangler rank-orders transform suggestions according to five criteria
  - The first three criteria rank transforms by their type; the remaining two rank transforms within types
    - **With type**
    - Firstly, explicit interactions are considered
    - Secondly, specification difficulty is considered
    - Thirdly, transform types are ranked based on their corpus frequency
    - **Within type**
    - First, transforms are sorted by frequency of equivalent transforms in the corpus
    - Second, transforms are sorted in ascending order using a measure of transform complexity

COMPARATIVE EVALUATION WITH EXCEL

- As an initial evaluation of Wrangler, a comparative user study with Microsoft Excel was conducted
- Subjects performed three common data cleaning tasks:
  - value extraction
  - missing value imputation
  - table reshaping
- The goal was to compare task completion times and observe data cleaning strategies
- The study showed that across all tasks, median performance in Wrangler was over twice as fast as Excel!
- This speed-up benefitted novice and expert Excel users alike

…continued



Task completion times – Wrangler vs Excel

Conclusion and Future Work

- We saw that novice Wrangler users can perform data cleaning tasks significantly faster than while using other famous tools like Excel

But still,

- People with highly specialized skills are spending more time than expected in “wrangling” tasks

So, the goal in the future is

- to introduce more research integrating methods from HCI, visualization, databases, and statistics to make data more accessible and informative

Thank you!