Advanced Data Management (CSCI 640/490)

Data Wrangling

Dr. David Koop

DataFrame Access and Manipulation

- `df.values → 2D NumPy array`
- Accessing a column:
  - `df["<column>"]`
  - `df.<column>`
  - Both return Series
  - Dot syntax only works when the column is a valid identifier
- Assigning to a column:

```
- df["<column>"] = <scalar> # all cells set to same value
- df["<column>"] = <array>  # values set in order
- df["<column>"] = <series> # values set according to match
                            # between df and series indexes
```

Indexing

- Same as with NumPy arrays but can use Series's index labels
- Slicing with labels: NumPy is **exclusive**, Pandas is **inclusive**!
  - ```
    s = Series(np.arange(4))
    s[0:2] # gives two values like numpy
    ```
  - ```
    s = Series(np.arange(4), index=['a', 'b', 'c', 'd'])
    s['a':'c'] # gives three values, not two!
    ```
- Obtaining data subsets
  - `[]`: get columns by label
  - `loc`: get rows/cols by label
  - `iloc`: get rows/cols by position (integer index)

- For single cells (scalars), also have `at` and `iat`

Filtering

- Same as with numpy arrays but allows use of column-based criteria
  - `data[data < 5] = 0`
  - `data[data['three'] > 5]`
  - `data < 5` → boolean data frame, can be used to select specific elements

Arithmetic

- Add, subtract, multiply, and divide are element-wise like numpy
- …but use labels to align
- …and missing labels lead to `NaN` (not a number) values

```
In [28]: obj3
Out[28]:
Ohio      35000
Oregon    16000
Texas     71000
Utah       5000
dtype: int64
```

```
In [29]: obj4
Out[29]:
California      NaN
Ohio          35000
Oregon        16000
Texas         71000
dtype: float64
```

```
In [30]: obj3 + obj4
Out[30]:
California        NaN
Ohio            70000
Oregon          32000
Texas          142000
Utah              NaN
dtype: float64
```

- also have `.add`, `.subtract`, … that allow `fill_value` argument
- `obj3.add(obj4, fill_value=0)`

Mutating Dataframes

- assign allows new columns to be created, returns "new" dataframe
  - `df2 = df.assign(Total=df.Points1 + df.Points2)`
- More reusable:
  - `df2 = df.assign(Total=lambda df: df.Points1 + df.Points2)`
- If you have columns that are not proper identifiers, can use **kwargs
  - `df2 = df.assign(**{"Total Points": lambda df: df.Points1 + df.Points2})`

Sorting by Value (sort_values)

- sort_values method on series
  - obj.sort_values()
- Missing values (NaN) are at the end by default (na_position controls, can be first)
- sort_values on DataFrame:
  - df.sort_values(<list-of-columns>)
  - df.sort_values(by=['a', 'b'])
  - Can also use axis=1 to sort by index labels

String Methods

- Can manipulate columns of strings
  - Use the `.str` modifier
- Most string and regex operations are available
- Examples:
  - `df.first_name.str.startswith("Jo")`
  - `df.name.str.split(' ')`

DuckDB

- SQL syntax with extras
  - `read_csv_auto`
  - similar string, datetime, array functions to pandas

Ibis

- More Pythonic interface to database or dataframe systems
- Uses DuckDB as default backend, can be configured to use others
- Syntax aligns better with SQL, potentially clearer
  - select
  - filter
  - mutate
  - group_by
  - order_by
  - unnest

Assignment 2

- Assignment 1 Questions with pandas, DuckDB, and Ibis
- CS 640 students do all, CS 490 do pandas & DuckDB (Ibis is EC)
- Can work by framework or by query
- Most questions can be answered with a single statement… but that statement can take a while to write
  - Read documentation
  - Check hints

Test 1

- Monday, Feb. 27
- In-class, 9:30-10:45am
- Format:
  - Multiple Choice
  - Free Response
- Information will be posted online

Statistics

- `sum`: column sums (`axis=1` gives sums over rows)
- missing values are excluded unless the whole slice is `NaN`
- `idxmax, idxmin` are like argmax, argmin (return index)
- `describe`: shortcut for easy stats!

```
In [204]: df.describe()
Out[204]:
            one       two
count  3.000000  2.000000
mean   3.083333 -2.900000
std    3.493685  2.262742
min    0.750000 -4.500000
25%    1.075000 -3.700000
50%    1.400000 -2.900000
75%    4.250000 -2.100000
max    7.100000 -1.300000
```

```
In [205]: obj = Series(['a', 'a', 'b', 'c'] * 4)

In [206]: obj.describe()
Out[206]:
count     16
unique     3
top        a
freq       8
dtype: object
```

Statistics

Method	Description
`count`	Number of non-NA values
`describe`	Compute set of summary statistics for Series or each DataFrame column
`min, max`	Compute minimum and maximum values
`argmin, argmax`	Compute index locations (integers) at which minimum or maximum value obtained, respectively
`idxmin, idxmax`	Compute index values at which minimum or maximum value obtained, respectively
`quantile`	Compute sample quantile ranging from 0 to 1
`sum`	Sum of values
`mean`	Mean of values
`median`	Arithmetic median (50% quantile) of values
`mad`	Mean absolute deviation from mean value
`var`	Sample variance of values
`std`	Sample standard deviation of values
`skew`	Sample skewness (3rd moment) of values
`kurt`	Sample kurtosis (4th moment) of values
`cumsum`	Cumulative sum of values
`cummin, cummax`	Cumulative minimum or maximum of values, respectively
`cumprod`	Cumulative product of values
`diff`	Compute 1st arithmetic difference (useful for time series)
`pct_change`	Compute percent changes

[W. McKinney, Python for Data Analysis]

Unique Values and Value Counts

- `unique` returns an array with only the unique values (no index)
  - ```
    s = Series(['c','a','d','a','a','b','b','c','c'])
    s.unique() # array(['c', 'a', 'd', 'b'])
    ```
- Data Frames use `drop_duplicates`
- `value_counts` returns a Series with index frequencies:
  - ```
    s.value_counts() # Series({'c': 3,'a': 3,'b': 2,'d': 1})
    ```

Handling Missing Data

Argument	Description
`dropna`	Filter axis labels based on whether values for each label have missing data, with varying thresholds for how much missing data to tolerate.
`fillna`	Fill in missing data with some value or using an interpolation method such as `'ffill'` or `'bfill'`.
`isnull`	Return like-type object containing boolean values indicating which values are missing / NA.
`notnull`	Negation of `isnull`.

[W. McKinney, Python for Data Analysis]

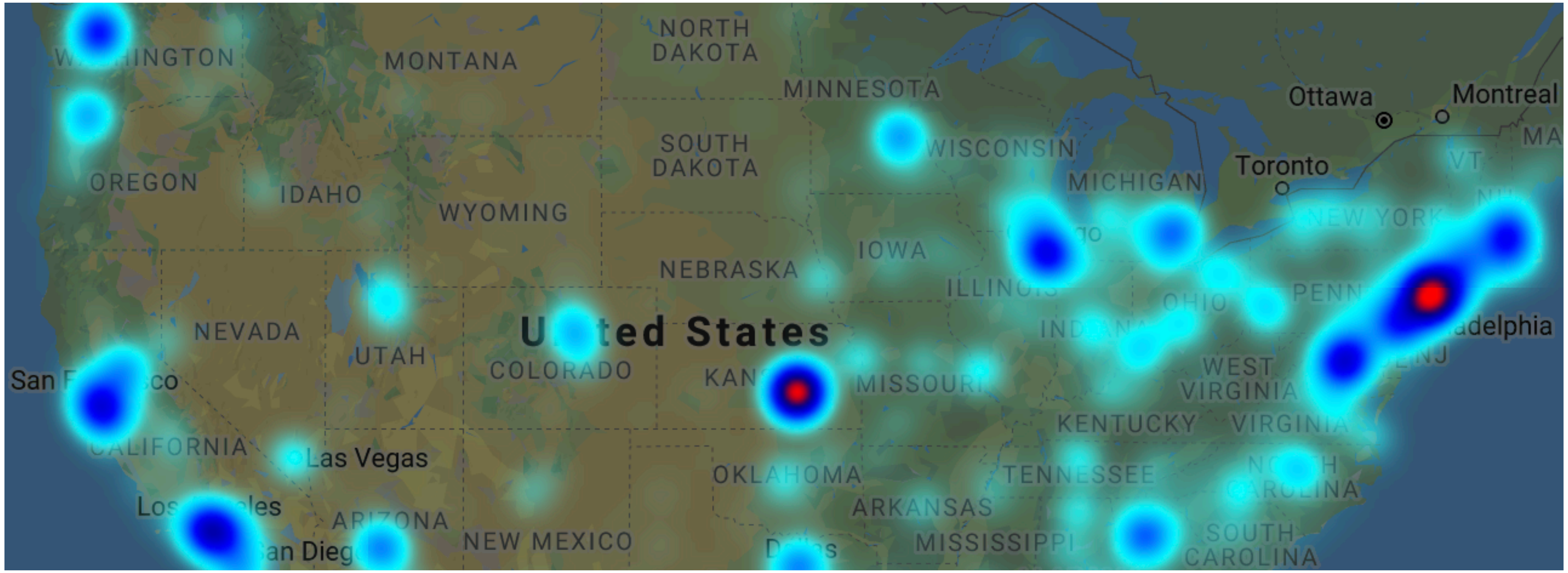
What if data isn't correct/trustworthy/in the right format?

Dirty Data

[Flickr]

Geolocation Errors

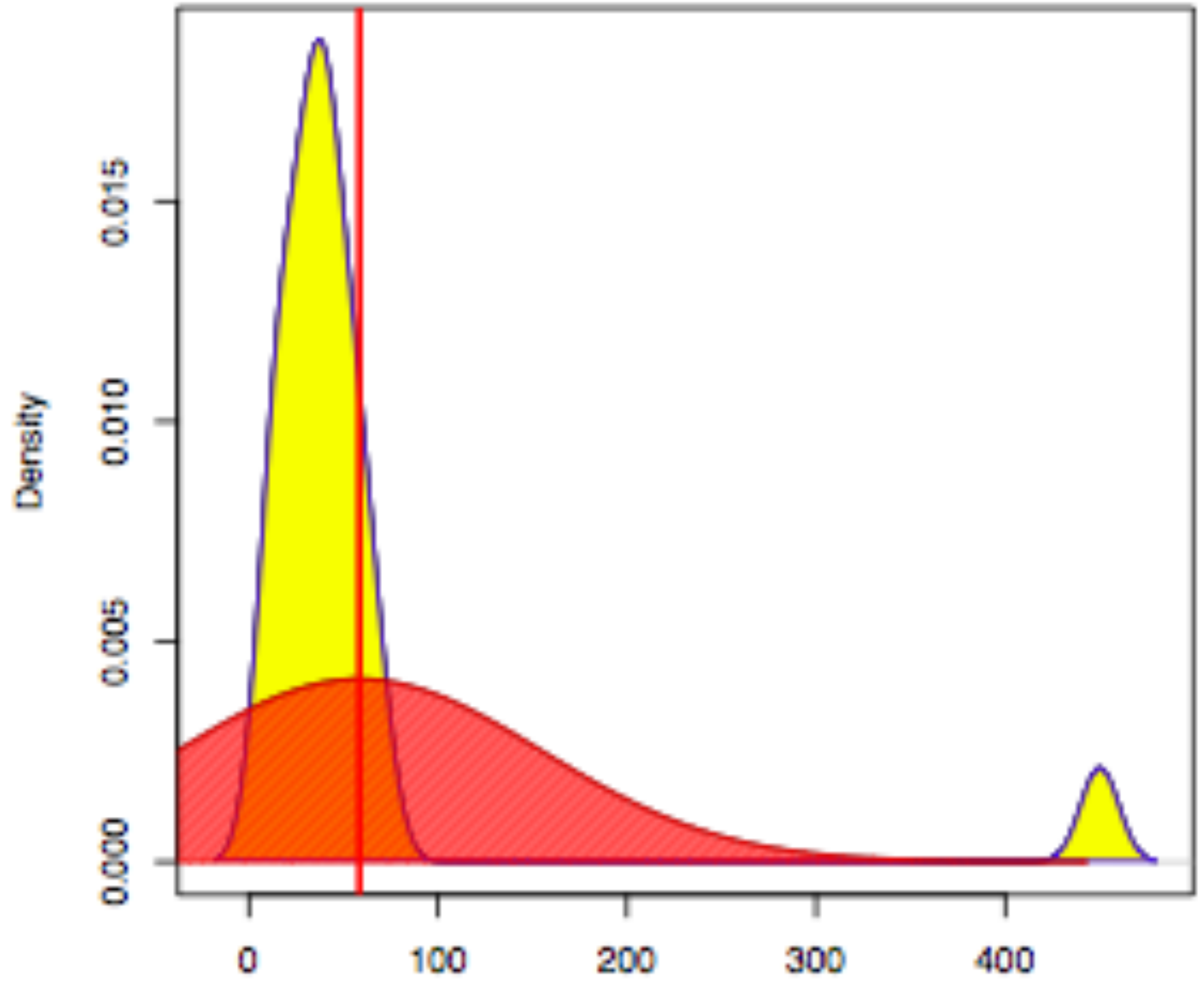
- Maxmind helps companies determine where users are located based on IP address
- "How a quiet Kansas home wound up with 600 million IP addresses and a world of trouble" [Washington Post, 2016]



Numeric Outliers

12	13	14	21	22	26	33	35	36	37	39	42	45	47	54	57	61	68	450

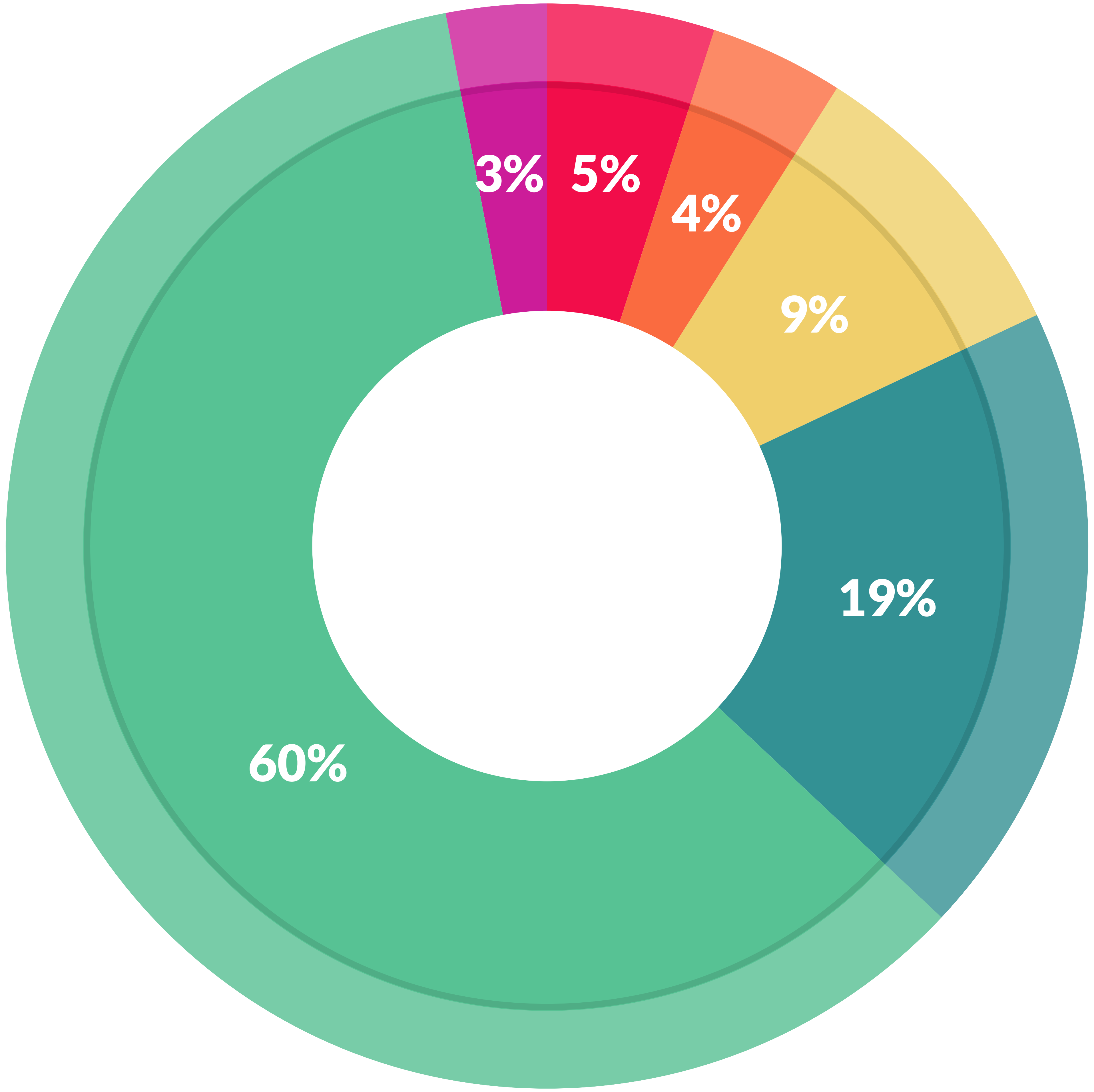
ages of employees (US)



- median 37
- mean 58.52632
- variance 9252.041

[J. Hellerstein via J. Canny et al.]

This takes a lot of time!

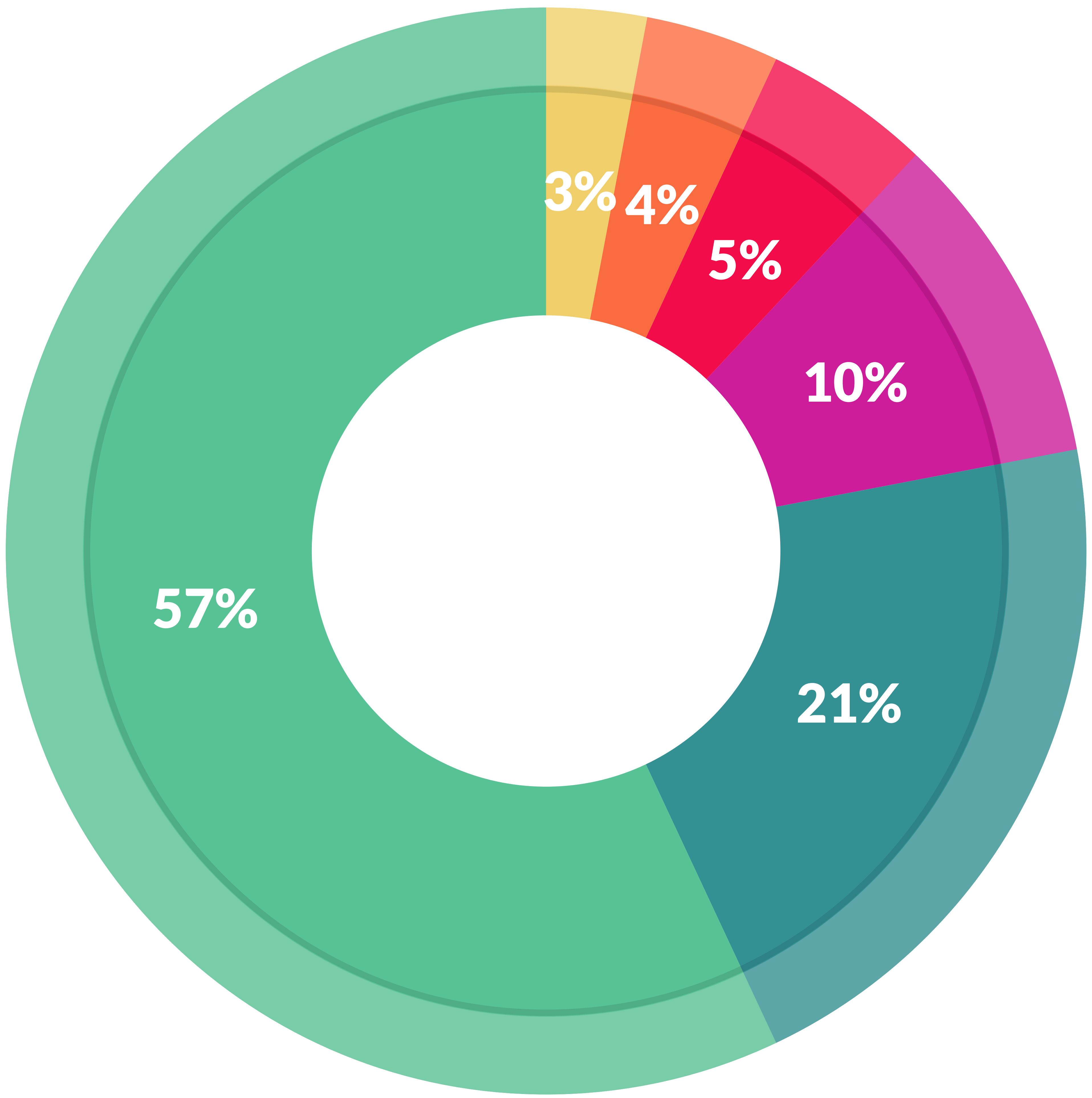


What data scientists spend the most time doing

- *Building training sets: 3%*
- *Cleaning and organizing data: 60%*
- *Collecting data sets; 19%*
- *Mining data for patterns: 9%*
- *Refining algorithms: 4%*
- *Other: 5%*

[CrowdFlower Data Science Report, 2016]

…and it isn't the most fun thing to do



What's the least enjoyable part of data science?

- *Building training sets: 10%*
- *Cleaning and organizing data: 57%*
- *Collecting data sets: 21%*
- *Mining data for patterns: 3%*
- *Refining algorithms: 4%*
- *Other: 5%*

[CrowdFlower Data Science Report, 2016]

Dirty Data: Statistician's View

- Some process produces the data
- Want a model but have non-ideal samples:
  - Distortion: some samples corrupted by a process
  - Selection bias: likelihood of a sample depends on its value
  - Left and right censorship: users come and go from scrutiny
  - Dependence: samples are not independent (e.g. social networks)
- You can add/augment models for different problems, but cannot model everything
- Trade-off between accuracy and simplicity

[J. Canny et al.]

Dirty Data: Database Expert's View

- Got a dataset
- Some values are missing, corrupted, wrong, duplicated
- Results are absolute (relational model)
- Better answers come from improving the quality of values in the dataset

[J. Canny et al.]

Dirty Data: Domain Expert's View

- Data doesn't look right
- Answer doesn't look right
- What happened?
- Domain experts carry an implicit model of the data they test against
- You don't always need to be a domain expert to do this
  - Can a person run 50 miles an hour?
  - Can a mountain on Earth be 50,000 feet above sea level?
  - Use common sense

[J. Canny et al.]

Dirty Data: Data Scientist's View

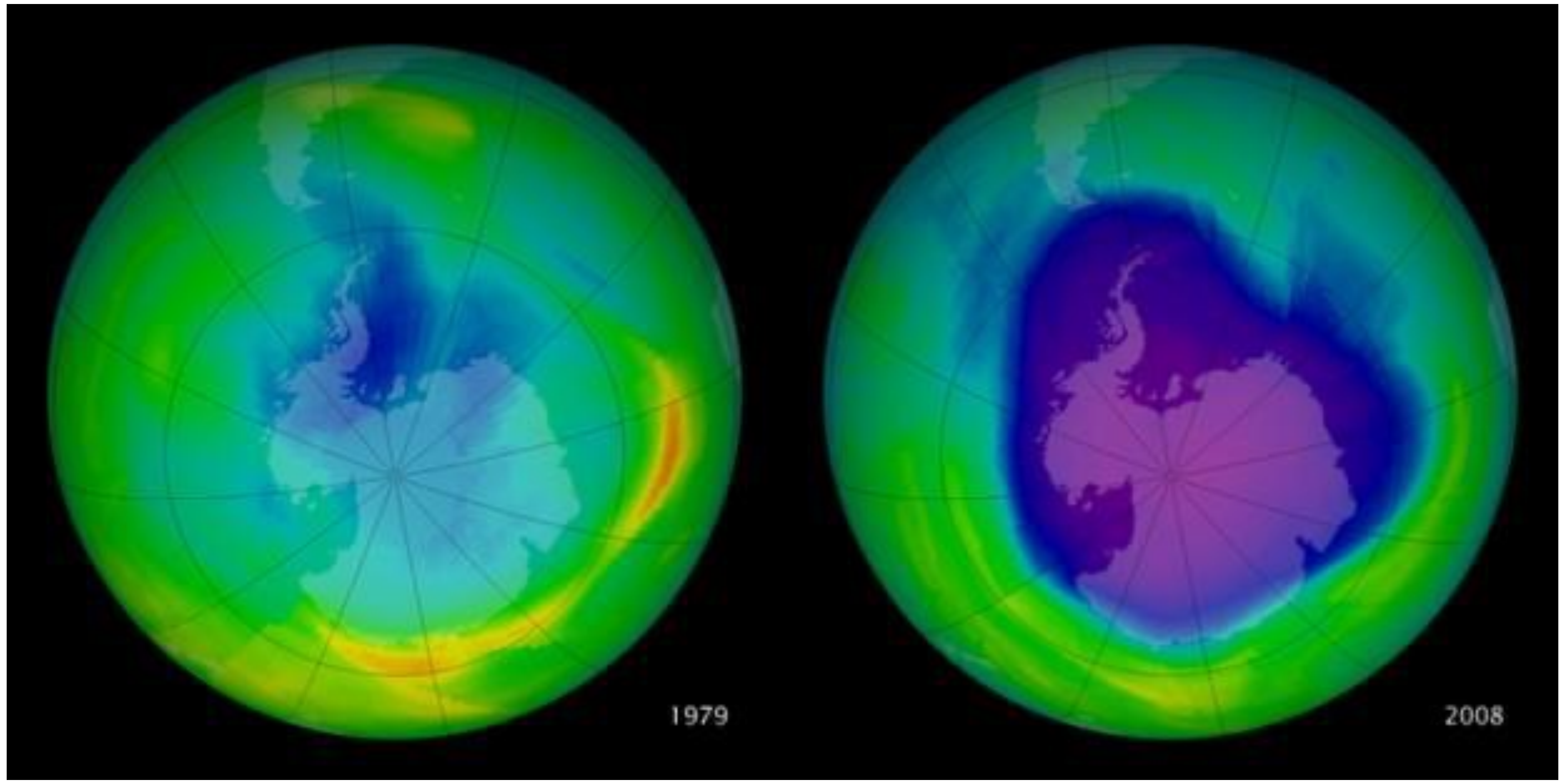
- Combination of the previous three views
- All of the views present problems with the data
- The goal may dictate the solutions:
  - Median value: don't worry too much about crazy outliers
  - Generally, aggregation is less susceptible by numeric errors
  - Be careful, the data may be correct…

[J. Canny et al.]

Be careful how you detect dirty data

- The appearance of a hole in the earth's ozone layer over Antarctica, first detected in 1976, was so unexpected that scientists didn't pay attention to what their instruments were telling them; they thought their instruments were malfunctioning.

  – National Center for Atmospheric Research



[Wikimedia]

Where does dirty data originate?

- Source data is bad, e.g. person entered it incorrectly
- Transformations corrupt the data, e.g. certain values processed incorrectly due to a software bug
- Integration of different datasets causes problems
- Error propagation: one error is magnified

[J. Canny et al.]

Types of Dirty Data Problems

- Separator Issues: e.g. CSV without respecting double quotes
  - `12, 13, "Doe, John", 45`
- Naming Conventions: `NYC` vs. `New York`
- Missing required fields, e.g. key
- Different representations: `2` vs. `two`
- Truncated data: `"Janice Keihanaikukauakahihuliheekahaunaele"` becomes `"Janice Keihanaikukauakahihuliheek"` on Hawaii license
- Redundant records: may be exactly the same or have some overlap
- Formatting issues: `2017-11-07` vs. `07/11/2017` vs. `11/07/2017`

[J. Canny et al.]

Data Wrangling

- Data wrangling: transform raw data to a more meaningful format that can be better analyzed
- Data cleaning: getting rid of inaccurate data
- Data transformations: changing the data from one representation to another
- Data reshaping: reorganizing the data
- Data merging: combining two datasets

Data Cleaning

Wrangler: Interactive Visual Specification of Data Transformation Scripts

S. Kandel, A. Paepcke, J. Hellerstein, J. Heer

Wrangler

- Data cleaning takes a lot of **time** and **human effort**
- "Tedium is the message"
- Repeating this process on multiple data sets is even worse!
- Solution:
  - interactive interface (mixed-initiative)
  - transformation language with natural language "translations"
  - suggestions + "programming by demonstration"

Your Critique/Questions

Example Critique

- Summary: Wrangler tackles data wrangling tasks by combining a language for specifying operations with an interface allowing users to specify the types of changes they are interested; the system can then generate suggested operations and demonstrates them on demand
- Critique: The suggestions may lead to states that a user cannot recover from easily. Suppose a suggestion looks like it works well, but a user later realizes was incorrect. They can backtrack, but it's often unclear where to and which other path to take. In addition, a user has to have some idea of the constructs of the language in order to edit parameters. Without a good idea of the impact of the parameters, the work may become as tedious as manual correction. Perhaps a more example-based strategy could help.

Previous Work: Potter's Wheel

- V. Raman and J. Hellerstein, 2001
- Defines structure extractions for identifying fields
- Defines transformations on the data
- Allows user interaction

Potter's Wheel: Structure Extraction

Example Column Value (Example erroneous values)	# Structures Enumerated	Final Structure Chosen (Punc = Punctuation)
-60	5	*Integer*
UNITED, DELTA, AMERICAN etc.	5	*IspellWord*
SFO, LAX etc. (JFK to OAK)	12	*AllCapsWord*
1998/01/12	9	*Int Punc(/) Int Punc(/) Int*
M, Tu, Thu etc.	5	*Capitalized Word*
06:22	5	*Int(len 2) Punc(:) Int(len 2)*
12.8.15.147 (ferret03.webtop.com)	9	*Double Punc('.') Double*
"GET\b (\b)	5	*Punc(") IspellWord Punc(\)*
/postmodern/lecs/xia/sld013.htm	4	ξ^*
HTTP	3	*AllCapsWord(HTTP)*
/1.0	6	*Punc(/) Double(1.0)*

[V. Raman and J. Hellerstein, 2001]

Potter's Wheel: Transforms

Transform	Definition		
Format	$\phi(R, i, f)$	$=$	$\{(a_1, \ldots, a_{i-1}, a_{i+1}, \ldots, a_n, f(a_i)) \mid (a_1, \ldots, a_n) \in R\}$
Add	$\alpha(R, x)$	$=$	$\{(a_1, \ldots, a_n, x) \mid (a_1, \ldots, a_n) \in R\}$
Drop	$\pi(R, i)$	$=$	$\{(a_1, \ldots, a_{i-1}, a_{i+1}, \ldots, a_n) \mid (a_1, \ldots, a_n) \in R\}$
Copy	$\kappa((a_1, \ldots, a_n), i)$	$=$	$\{(a_1, \ldots, a_n, a_i) \mid (a_1, \ldots, a_n) \in R\}$
Merge	$\mu((a_1, \ldots, a_n), i, j, \text{glue})$	$=$	$\{(a_1, \ldots, a_{i-1}, a_{i+1}, \ldots, a_{j-1}, a_{j+1}, \ldots, a_n, a_i \oplus \text{glue} \oplus a_j) \mid (a_1, \ldots, a_n) \in R\}$
Split	$\omega((a_1, \ldots, a_n), i, \text{splitter})$	$=$	$\{(a_1, \ldots, a_{i-1}, a_{i+1}, \ldots, a_n, \text{left}(a_i, \text{splitter}), \text{right}(a_i, \text{splitter})) \mid (a_1, \ldots, a_n) \in R\}$
Divide	$\delta((a_1, \ldots, a_n), i, \text{pred})$	$=$	$\{(a_1, \ldots, a_{i-1}, a_{i+1}, \ldots, a_n, a_i, \text{null}) \mid (a_1, \ldots, a_n) \in R \wedge \text{pred}(a_i)\} \cup$ $\{(a_1, \ldots, a_{i-1}, a_{i+1}, \ldots, a_n, \text{null}, a_i) \mid (a_1, \ldots, a_n) \in R \wedge \neg\text{pred}(a_i)\}$
Fold	$\lambda(R, i_1, i_2, \ldots i_k)$	$=$	$\{(a_1, \ldots, a_{i_1-1}, a_{i_1+1}, \ldots, a_{i_2-1}, a_{i_2+1}, \ldots, a_{i_k-1}, a_{i_k+1}, \ldots, a_n, a_{i_l}) \mid$ $(a_1, \ldots, a_n) \in R \wedge 1 \leq l \leq k\}$
Select	$\sigma(R, \text{pred})$	$=$	$\{(a_1, \ldots, a_n) \mid (a_1, \ldots, a_n) \in R \wedge \text{pred}((a_1, \ldots, a_n))\}$

Notation: R is a relation with n columns. i, j are column indices and a_i represents the value of a column in a row. x and glue are values. f is a function mapping values to values. $x \oplus y$ concatenates x and y. splitter is a position in a string or a regular expression, left$(x, \text{splitter})$ is the left part of x after splitting by splitter. pred is a function returning a boolean.

[V. Raman and J. Hellerstein, 2001]

Potter's Wheel: Example

		Stewart,Bob
Anna	Davis	
		Dole,Jerry
Joan	Marsh	

Format
'(.*), (.*)' to '\2 \1'

		Bob Stewart
Anna	Davis	
		Jerry Dole
Joan	Marsh	

Split at ' '

		Bob	Stewart
Anna	Davis		
		Jerry	Dole
Joan	Marsh		

2 Merges

Bob	Stewart
Anna	Davis
Jerry	Dole
Joan	Marsh

[V. Raman and J. Hellerstein, 2001]

Potter's Wheel: Inferring Structure from Examples

Example Values Split By User (| is user specified split position)	Inferred Structure	Comments
Taylor, Jane |, \$52,072 Blair, John |, \$73,238 Tony Smith |, \$1,00,533	$(< \xi^* > <$ ',' *Money* $>)$	Parsing is doable despite no good delimiter. A *regular expression* domain can infer a structure of $[0-9,]* for last component.
MAA |to| SIN JFK |to| SFO LAX |–| ORD SEA |/| OAK	($<$*len 3 identifier*$>$ $< \xi^* >$ $<$ *len 3 identifier*$>$)	Parsing is possible despite multiple delimiters.
321 Blake #7 |, Berkeley |, CA 94720 719 MLK Road |, Fremont |, CA 95743	($<$*number* $\xi^* > <$ ',' *word*$>$ $<$',' *(2 letter word) (5 letter integer)*$>$)	Parsing is easy because of consistent delimiter.

[V. Raman and J. Hellerstein, 2001]

Wrangler Transformation Language

- Based on Potter's Wheel
- Map: Delete, Extract, Cut, Split, Update
- Lookup/join: Use external data (e.g. from zipcode→state)
- Reshape: Fold and Unfold (aka pivot)
- Positional: Fill and lag
- Sorting, aggregation, key generation, schema transforms

Interface

- Automated Transformation Suggestions
- Editable Natural Language Explanations
    - Fill **Bangladesh** by **copying** values from **above**
    - Fill **Bangladesh** by [averaging / ✓ copying / interpolating] values from **above**
    - Fill **Bangladesh** by **averaging** the **5** values from **above**

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

- Visual Transformation Previews
- Transformation History

[S. Kandel et al., 2011]

Automation from past actions

- Infer parameter sets from user interaction
- Generating transforms
- Ranking and ordering transformations:
  - Based on user preferences, difficulty, and corpus frequency
  - Sort transforms by type and diversify suggestions

(a) `Reported crime in Alabama`

(b)
before: {'in', ' '} — 'Alabama' → {'Alabama', *word*}
selection: {'Alabama'} — 'in' → {'in', *word*, *lowercase*}
after: ∅ — ' ' → {' '}

(c)
before: {(' '), ('in', ' '), (*word*, ' '), (*lowercase*, ' ')}
selection: {('Alabama'), (*word*)}
after: ∅

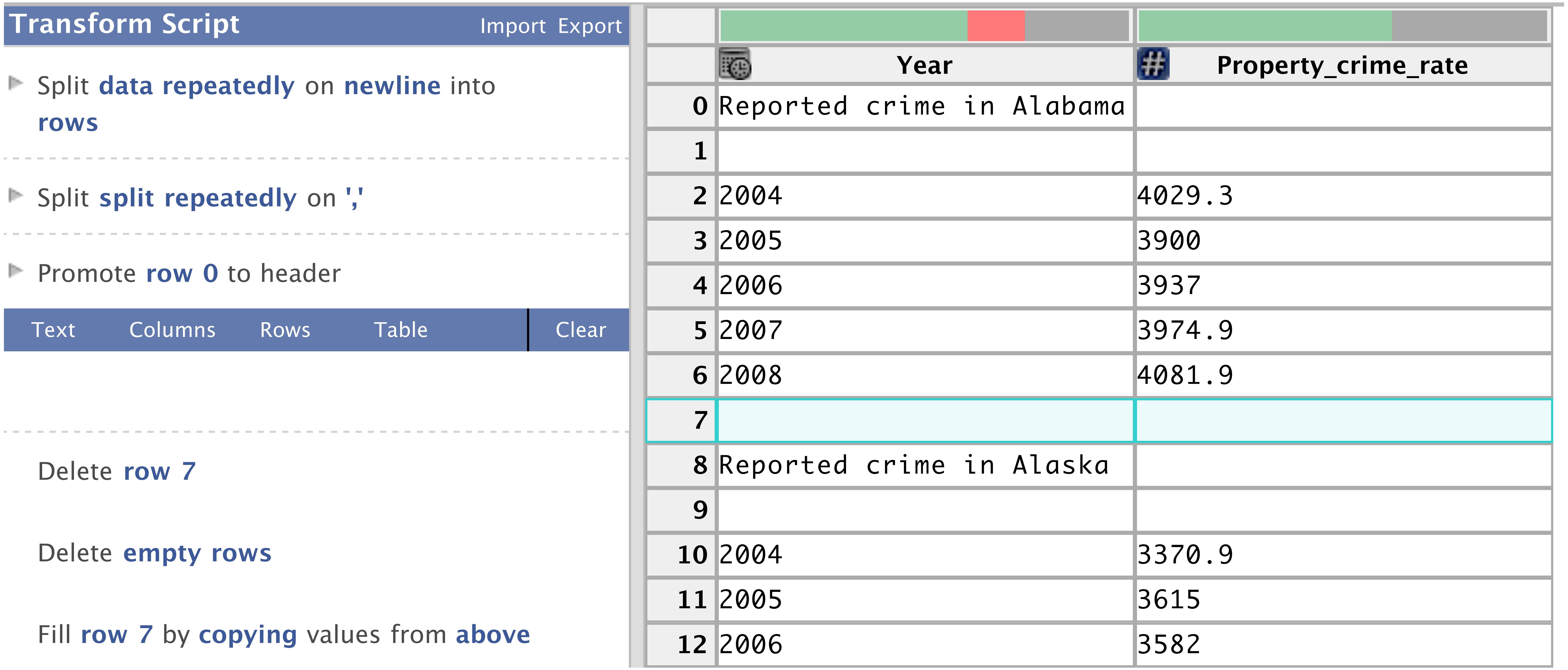
(d)
{(),('Alabama'),()} — ~~{(),(*word*),()}~~
~~{(' '),(),()}~~ — ~~{(*word*, ' '),(),()}~~
{(' '),('Alabama'),()} — {(*word*, ' '),('Alabama'),()}
~~{(' '),(*word*),()}~~ — ~~{(*word*, ' '),(*word*),()}~~
{('in', ' '),(),()} — {(*lowercase*, ' '),(),()}
{('in', ' '),('Alabama'),()} — {(*lowercase*, ' '),('Alabama'),()}
{('in', ' '),(*word*),()} — ~~{(*lowercase*, ' '),(*word*),()}~~

(e) {(*lowercase*, ' '),('Alabama'),()} → /[a-z]+ (Alabama)/

[S. Kandel et al., 2011]

Data Wrangler Demo

- http://vis.stanford.edu/wrangler/app/



	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