Clean Your Data

Dr. Bo (Beth) Sun

Data Cleaning

Why data can be dirty?

How dirty is real data?

Examples

- Jan 19, 2016
- January 19, 16
- 1/19/16
- 2006-01-19
- 19/1/16

How dirty is real data?

Discuss with you neighbors (group of 2-3)

90 seconds

Comes up with **5+ kinds of “data dirtiness”**

How dirty is real data?

Examples

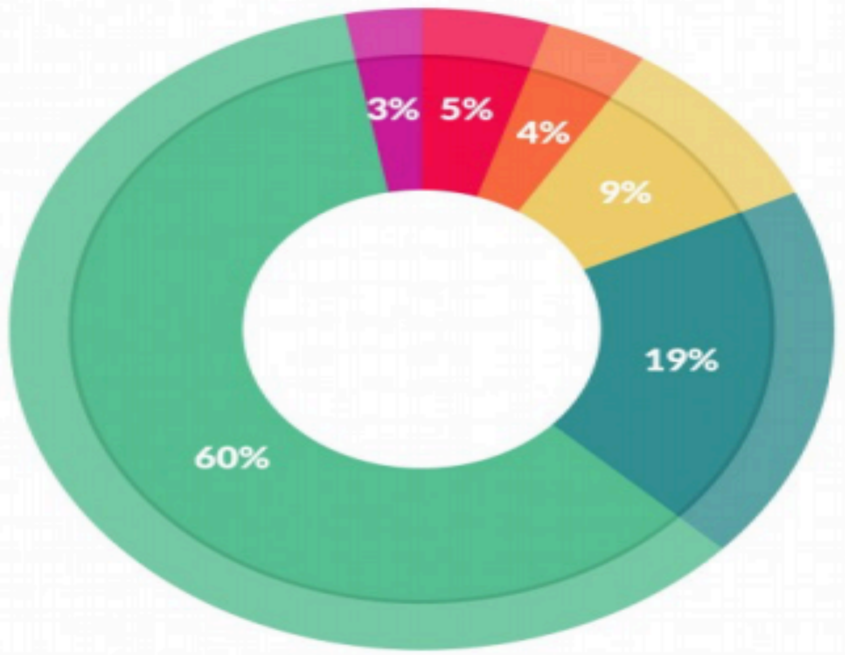
- unbalanced/“outliers”
- leading zeros…
- different units/measurements (pounds…)
- missing data
- spelling errors
- wrong data types
- cases lower/upper
- data in the wrong place (shifted somehow)
- file format
- inconsistent (last name/first name order exchange)
- encoding errors
- duplication
- different representation for the same thing
- different scales

What is Messy Data?

2015-10-14	$1,000	ID
10/14/2015	1000	I.D.
10/14/15	1,000	US-ID
Oct 14, 2015	1000 dollars	idaho
Wed, Oct 14th	US$1000	Idaho,
42291	$1k	Ihaho

“80%” Time Spent on Data Preparation

Cleaning Big Data: Most Time-Consuming, Least Enjoyable Data Science Task, Survey Says [Forbes]
http://www.forbes.com/sites/gilpress/2016/03/23/data-preparation-most-time-consuming-least-enjoyable-data-science-task-survey-says/#73bf5b137f75



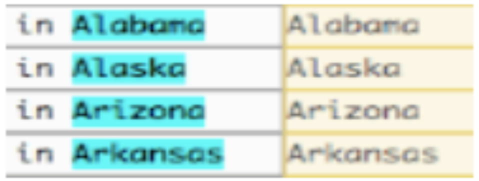
What data scientists spend the most time doing

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

Data Cleaners

Watch videos

- Data Wrangler (research at Stanford)
- Open Refine (previously **Google Refine**)



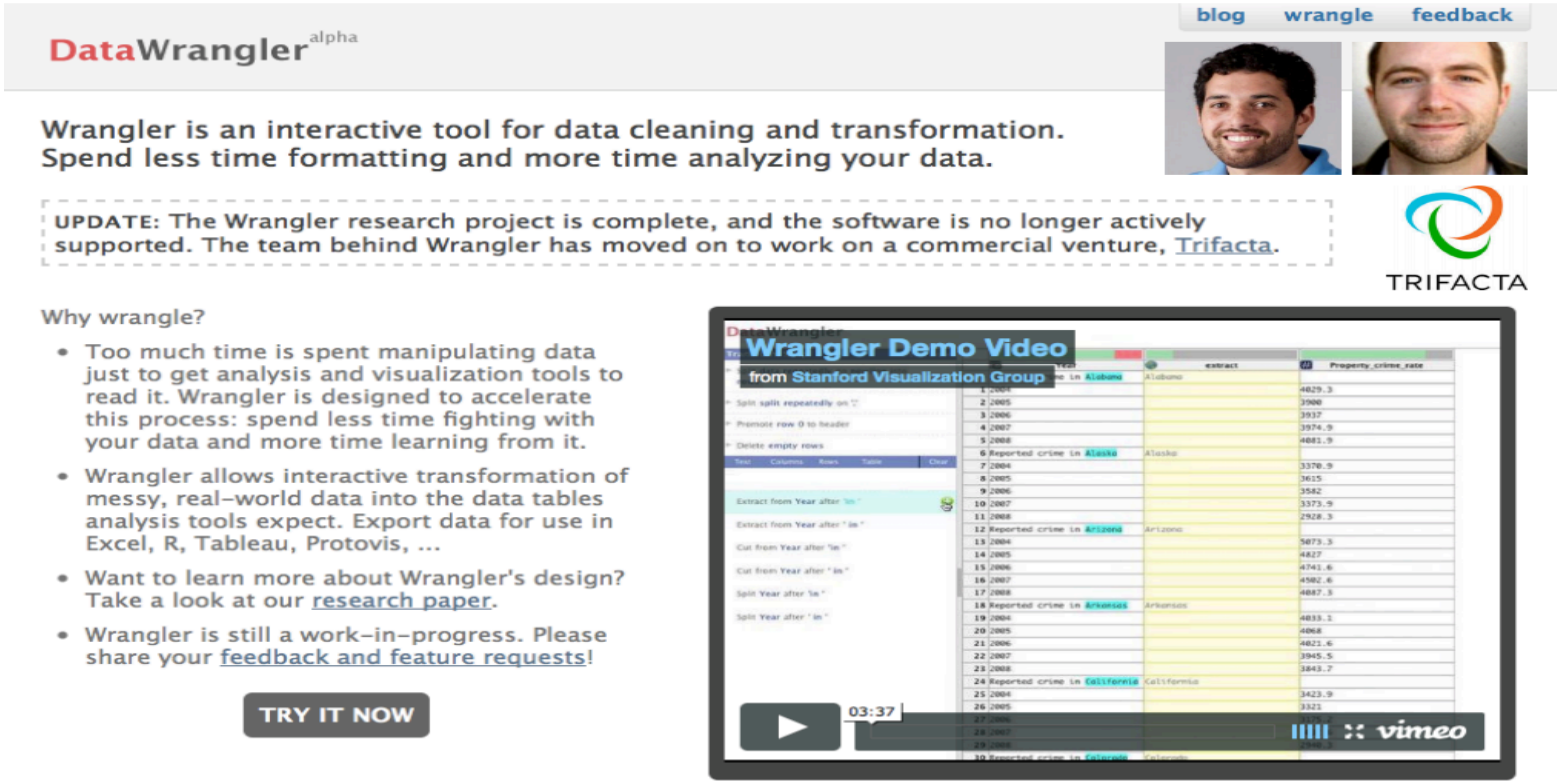
in Alabama	Alabama
in Alaska	Alaska
in Arizona	Arizona
in Arkansas	Arkansas

Write down

- Examples of **data dirtiness**
- Tool's **features** demo-ed (or that you like)

Will collectively summarize similarities and differences afterwards

Open Refine: http://openrefine.org
Data Wrangler: http://vis.stanford.edu/wrangler/



DataWrangler alpha

blog wrangle feedback

Wrangler is an interactive tool for data cleaning and transformation.
Spend less time formatting and more time analyzing your data.

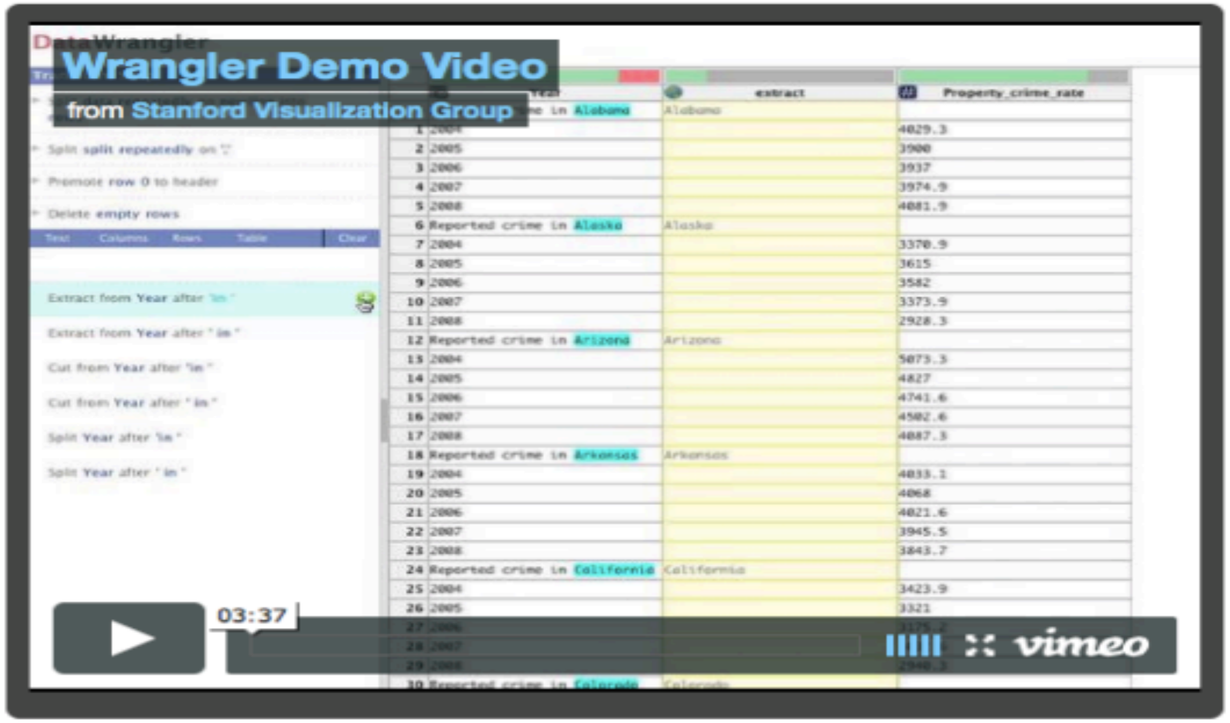
UPDATE: The Wrangler research project is complete, and the software is no longer actively supported. The team behind Wrangler has moved on to work on a commercial venture, Trifacta.

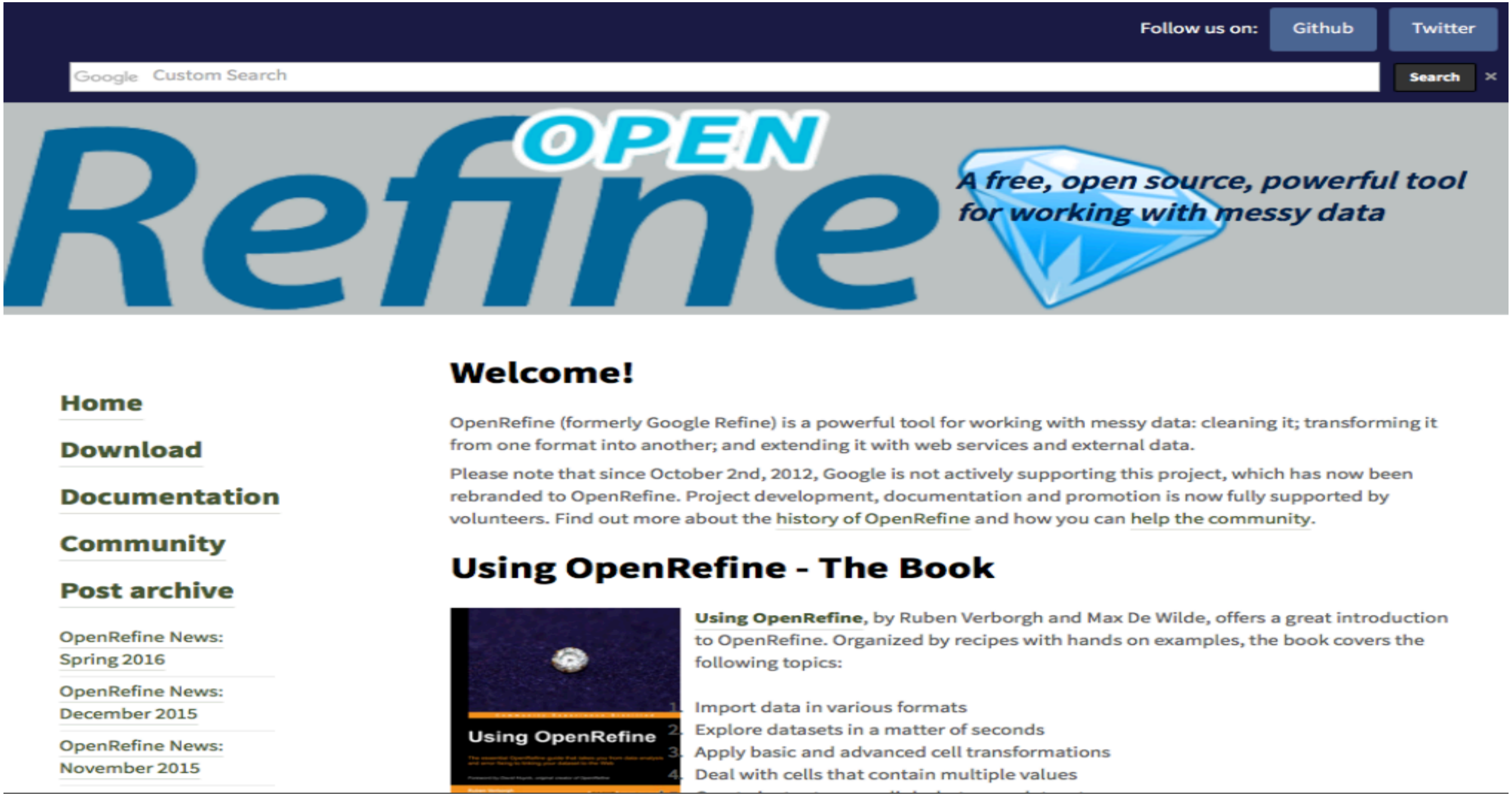
TRIFACTA

Why wrangle?

- Too much time is spent manipulating data just to get analysis and visualization tools to read it. Wrangler is designed to accelerate this process: spend less time fighting with your data and more time learning from it.
- Wrangler allows interactive transformation of messy, real-world data into the data tables analysis tools expect. Export data for use in Excel, R, Tableau, Protovis, ...
- Want to learn more about Wrangler's design? Take a look at our research paper.
- Wrangler is still a work-in-progress. Please share your feedback and feature requests!

TRY IT NOW





Home

Download

Documentation

Community

Post archive

OpenRefine News: Spring 2016

OpenRefine News: December 2015

OpenRefine News: November 2015

Welcome!

OpenRefine (formerly Google Refine) is a powerful tool for working with messy data: cleaning it; transforming it from one format into another; and extending it with web services and external data.

Please note that since October 2nd, 2012, Google is not actively supporting this project, which has now been rebranded to OpenRefine. Project development, documentation and promotion is now fully supported by volunteers. Find out more about the history of OpenRefine and how you can help the community.

Using OpenRefine - The Book

Using OpenRefine, by Ruben Verborgh and Max De Wilde, offers a great introduction to OpenRefine. Organized by recipes with hands on examples, the book covers the following topics:

1. Import data in various formats
2. Explore datasets in a matter of seconds
3. Apply basic and advanced cell transformations
4. Deal with cells that contain multiple values

What can the tools do?

identify similar words

history (in W: script)

clustering

show the “impact” of changes (#rows changed)

transformation

offline processing

highlight outliers

“pivoting”

previewing

extract part of word, do some generalization

histogram/vis for each column (highlight missing values)

suggested operations

O = Open Refine

W = Data wrangler

Refine Use Cases:

Clean - discover and fix inconsistency with faceting, clustering, cell transforms, GREL expressions...

Transform - change formats or reshape with split/join multi valued cells, split columns, transpose columns/rows...

Extend - enrich data by combining files, merging projects, fetching URLs, reconciliation with online databases...

Automate - reuse your processing routine by exporting operation history in JSON!

OpenRefine Expression Language *(GREL)*

- Example

ID	Friend	Age
1.	John Smith	28
2.	Jane Doe	33

- value.split(" ")[1] == “Smith”; “Doe”. *What about value.split(" ")[0]?*

https://github.com/OpenRefine/OpenRefine/wiki/Understanding-Expressions

GREL Basic

example	description	If value = “ hello OpenRefine”
value + " (approved)"	concatenate two strings; whatever is in value gets converted to a string first	“ hello OpenRefine (approved)”
value + 2.239	add two numbers; if value actually holds something other than a number, this becomes a string concatenation	“ hello OpenRefine2.239”
value.trim().length()	trimming leading and trailing whitespace of value and than take the length of the result	16
value.substring(7, 10)	take the substring of value from character index 7 up to and excluding character index 10	“ Op”
value.substring(13)	take the substring of value from character index 13 until the end of the string	“efine”

Function Syntax

- functionName (arg0, arg1, ...)
- arg0.functionName(arg1, ...) May not work with OpenRefine 3.0

dot shorthand notation	full notation
value.trim().length()	length(trim(value))
value.substring(7, 10)	substring(value, 7, 10)
value.substring(13)	substring(value, 13)

Array Syntax

example	**description**
value[1,3]	access the substring of value starting from character index 1 up to but excluding character index 3
"internationalization"[1,3]	return nt
"internationalization"[1,-2]	return nternationalizati (negative indexes are counted from the end)

Controls

example	description
if(value.length() > 10, "big string", "small string")	if the length of what value stands for is great than 10 characters, then return "big string", otherwise, return "small string"
if(mod(row.index, 2) == 0, "even", "odd")	if the 2 modulus of the row index is zero, then output "even", otherwise, output "odd"
forEach("Once upon a time in Mexico".split(" "), v, v.length())	return array of lengths of words, [4, 4, 1, 4, 2, 6]

```
if ( test_condition , true_result , false_result )
```

```
for ( array_subexpr , element_var_name , element_subexpr )
```

More Controls

Compute average word length for the string "Once upon a time in Mexico”:

```
forEach("Once upon a time in Mexico".split(" "), v, v.length()).sum() / "Once upon a time in Mexico".split("
").length()
```

⇕

```
with("Once upon a time in Mexico".split(" "), a, forEach(a, v, v.length()).sum() / a.length())
```

```
with ( subexpr1 , var_name , subexpr2 )
```

GREL String Functions

- Pls reference https://github.com/OpenRefine/OpenRefine/wiki/GREL-String-Functions
    - replace
    - match
    - partition
    - split

GREL Boolean Functions

and(boolean b1, boolean b2, ...etc)

Logically AND two or more booleans to yield a boolean. For example, and(1 < 3, 1 > 0)returns true because both conditions are true.

or(boolean b1, boolean b2, ...etc)

Logically OR two or more booleans to yield a boolean. For example, or(1 < 3, 1 > 7) returns true because at least one of the conditions (the first one) is true.

not(boolean b)

Logically NOT a boolean to yield another boolean. For example, not(1 > 7) returns truebecause 1 > 7 itself is false.

xor(boolean b1, boolean b2, ...etc)

Logically XOR (exclusive-or) two or more booleans to yield a boolean. For example, xor(1 < 3, 1 > 7) returns true because only one of the conditions (the first one) is true. xor(1 < 3, 1 < 7) returns false because more than one of the conditions is true.

https://github.com/OpenRefine/OpenRefine/wiki/GREL-Boolean-Functions

All GREL Functions

- Pls refer https://github.com/OpenRefine/OpenRefine/wiki/GREL-Functions

Understanding Regular Expressions

1. \d, same as [0-9]
2. \D, same as [^0-9], meaning any character other than a digit
3. \s, meaning any whitespace character (space, tab, new line, carriage return)
4. \S, meaning any character other than whitespace characters
5. \w, same as [a-zA-Z_0-9], meaning any character that can be part of a word (where the meaning of "word" is more liberal than an English word)
6. \W, meaning any non-word character
7. . meaning any single character
8. [ ] one of the characters inside
9. \+ occurring one or more times
10. ? an optional character or group of characters

- . Must be written as \.
- \+ must be written as \+
- ? must be written as \?
- [must be written as \[

Examples

How to form a RegExp for decimal numbers?

- 120.0232898
- -0.2909
- +2.343

We can describe a decimal number as follows:

- It optionally starts with a sign (minus or plus)
- Then it consists of a sequence of one or more digits
- Then optionally if it has a decimal part, it contains
    - a period, followed by
    - one or more digits

```
[-+]?[0-9]+(\.[0-9]+)?
```