Reg(ular)?Ex(pressions)?

A Multi-Tool for Working with Text

James Truitt

BitCurator Forum Philadelphia Satellite

2024-03-19

2/14/1865

5/22/1890

7/4/1878

9/30/1899

3/15/1885

1865-02-14

1890-05-22

1878-07-04

1899-09-30

1885-03-15

Keeran, Vincent What Is Causing the Current Rise In Prices 10/12/47

Keeran, Vincent.
What Is Causing the Current Rise In Prices?
1947-10-12

personal|primary|naf|http://id.loc.gov/authorities/names/|http://id.loc.gov/authorities/names/n85329150|Lockwood, Belva Ann, 1830-1917|text|marcrelator|creator

relators:cre:person:Lockwood, Belva Ann, 1830-1917

What are Regular Expressions?

A language for pattern matching

Why Use Regular Expressions?

- Flexible
- Powerful
- Available
- Output-focused

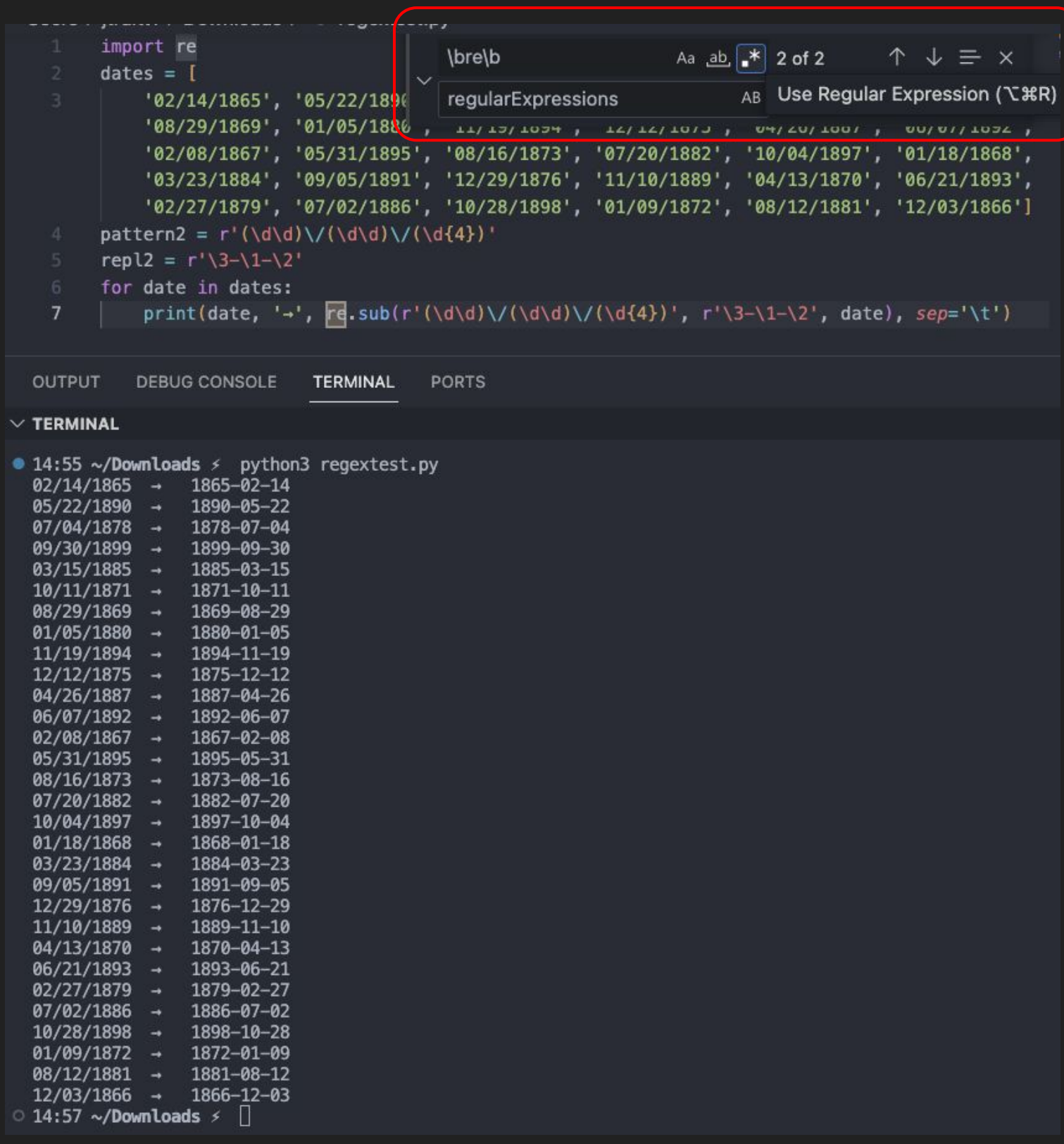
Where to Use Regular Expressions?

```python
import re
dates = [
    '02/14/1865', '05/22/1890', '07/04/1878', '09/30/1899', '03/15/1885', '10/11/1871',
    '08/29/1869', '01/05/1880', '11/19/1894', '12/12/1875', '04/26/1887', '06/07/1892',
    '02/08/1867', '05/31/1895', '08/16/1873', '07/20/1882', '10/04/1897', '01/18/1868',
    '03/23/1884', '09/05/1891', '12/29/1876', '11/10/1889', '04/13/1870', '06/21/1893',
    '02/27/1879', '07/02/1886', '10/28/1898', '01/09/1872', '08/12/1881', '12/03/1866']
pattern2 = r'(\d\d)\/(\d\d)\/(\d{4})'
repl2 = r'\3-\1-\2'
for date in dates:
    print(date, '→', re.sub(r'(\d\d)\/(\d\d)\/(\d{4})', r'\3-\1-\2', date), sep='\t')
```

OUTPUT DEBUG CONSOLE TERMINAL PORTS

TERMINAL

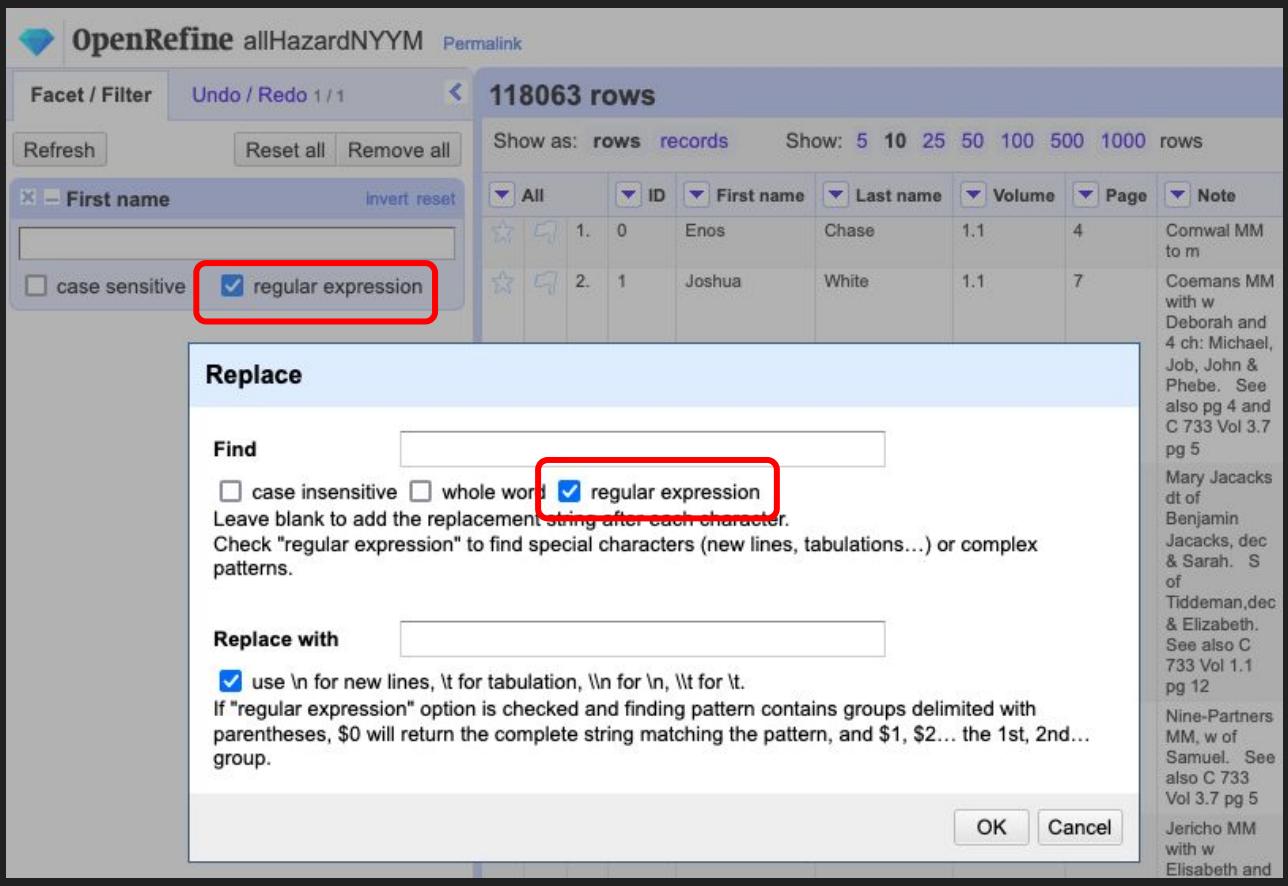
```
14:55 ~/Downloads ⚡  python3 regextest.py
02/14/1865  →   1865-02-14
05/22/1890  →   1890-05-22
07/04/1878  →   1878-07-04
09/30/1899  →   1899-09-30
03/15/1885  →   1885-03-15
10/11/1871  →   1871-10-11
08/29/1869  →   1869-08-29
01/05/1880  →   1880-01-05
11/19/1894  →   1894-11-19
12/12/1875  →   1875-12-12
04/26/1887  →   1887-04-26
06/07/1892  →   1892-06-07
02/08/1867  →   1867-02-08
05/31/1895  →   1895-05-31
08/16/1873  →   1873-08-16
07/20/1882  →   1882-07-20
10/04/1897  →   1897-10-04
01/18/1868  →   1868-01-18
03/23/1884  →   1884-03-23
09/05/1891  →   1891-09-05
12/29/1876  →   1876-12-29
11/10/1889  →   1889-11-10
04/13/1870  →   1870-04-13
06/21/1893  →   1893-06-21
02/27/1879  →   1879-02-27
07/02/1886  →   1886-07-02
10/28/1898  →   1898-10-28
01/09/1872  →   1872-01-09
08/12/1881  →   1881-08-12
12/03/1866  →   1866-12-03
14:57 ~/Downloads ⚡
```

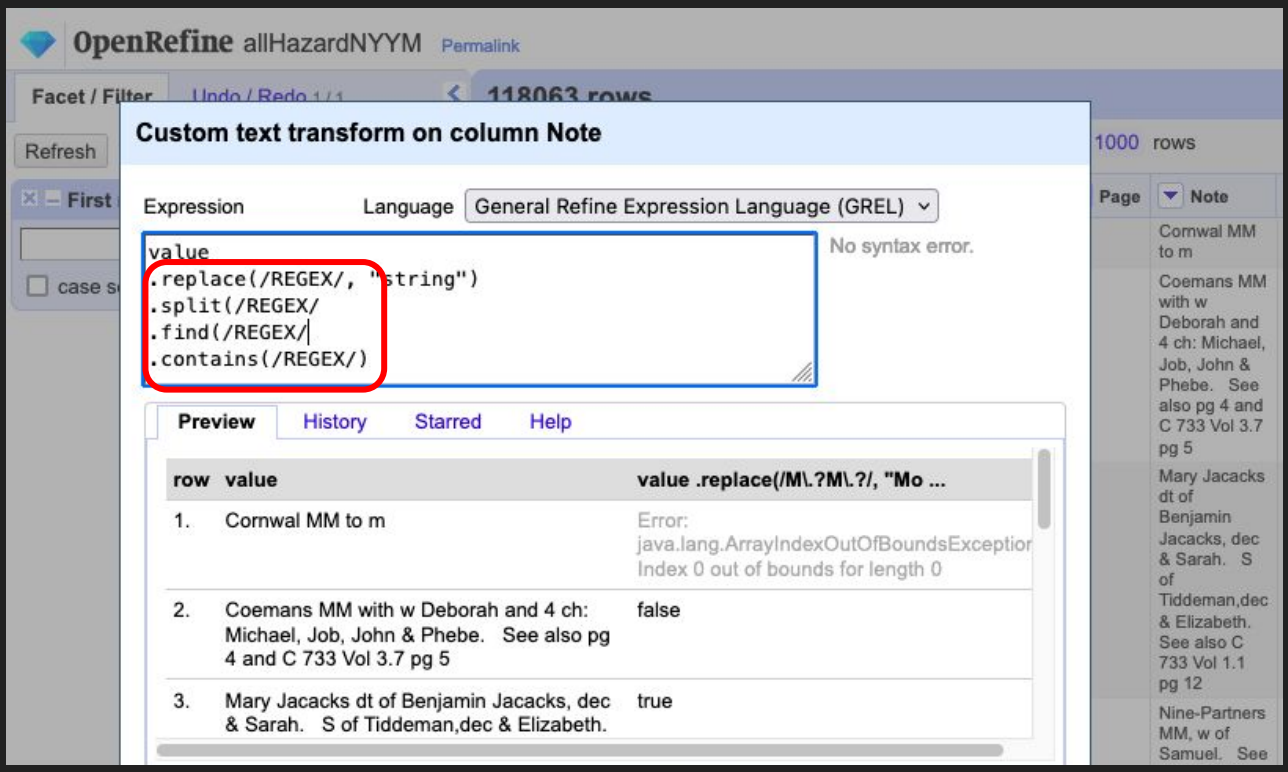

import re
dates = [
'02/14/1865', '05/22/1890', '07/04/1878', '09/30/1899', '03/15/1885', '10/11/1871',
'08/29/1869', '01/05/1880', '11/19/1894', '12/12/1875', '04/26/1887', '06/07/1892',
'02/08/1867', '05/31/1895', '08/16/1873', '07/20/1882', '10/04/1897', '01/18/1868',
'03/23/1884', '09/05/1891', '12/29/1876', '11/10/1889', '04/13/1870', '06/21/1893',
'02/27/1879', '07/02/1886', '10/28/1898', '01/09/1872', '08/12/1881', '12/03/1866']
pattern2 = r'(\d\d)\/(\d\d)\/(\d{4})'
repl2 = r'\3-\1-\2'
for date in dates:
print(date, '→', re.sub(r'(\d\d)\/(\d\d)\/(\d{4})', r'\3-\1-\2', date), sep='\t')
\bre\b
Aa
ab
.*
2 of 2
regularExpressions
AB
Use Regular Expression (⌥⌘R)
OUTPUT
DEBUG CONSOLE
TERMINAL
PORTS
TERMINAL
14:55 ~/Downloads ⚡ python3 regextest.py
02/14/1865 → 1865-02-14
05/22/1890 → 1890-05-22
07/04/1878 → 1878-07-04
09/30/1899 → 1899-09-30
03/15/1885 → 1885-03-15
10/11/1871 → 1871-10-11
08/29/1869 → 1869-08-29
01/05/1880 → 1880-01-05
11/19/1894 → 1894-11-19
12/12/1875 → 1875-12-12
04/26/1887 → 1887-04-26
06/07/1892 → 1892-06-07
02/08/1867 → 1867-02-08
05/31/1895 → 1895-05-31
08/16/1873 → 1873-08-16
07/20/1882 → 1882-07-20
10/04/1897 → 1897-10-04
01/18/1868 → 1868-01-18
03/23/1884 → 1884-03-23
09/05/1891 → 1891-09-05
12/29/1876 → 1876-12-29
11/10/1889 → 1889-11-10
04/13/1870 → 1870-04-13
06/21/1893 → 1893-06-21
02/27/1879 → 1879-02-27
07/02/1886 → 1886-07-02
10/28/1898 → 1898-10-28
01/09/1872 → 1872-01-09
08/12/1881 → 1881-08-12
12/03/1866 → 1866-12-03
14:57 ~/Downloads ⚡

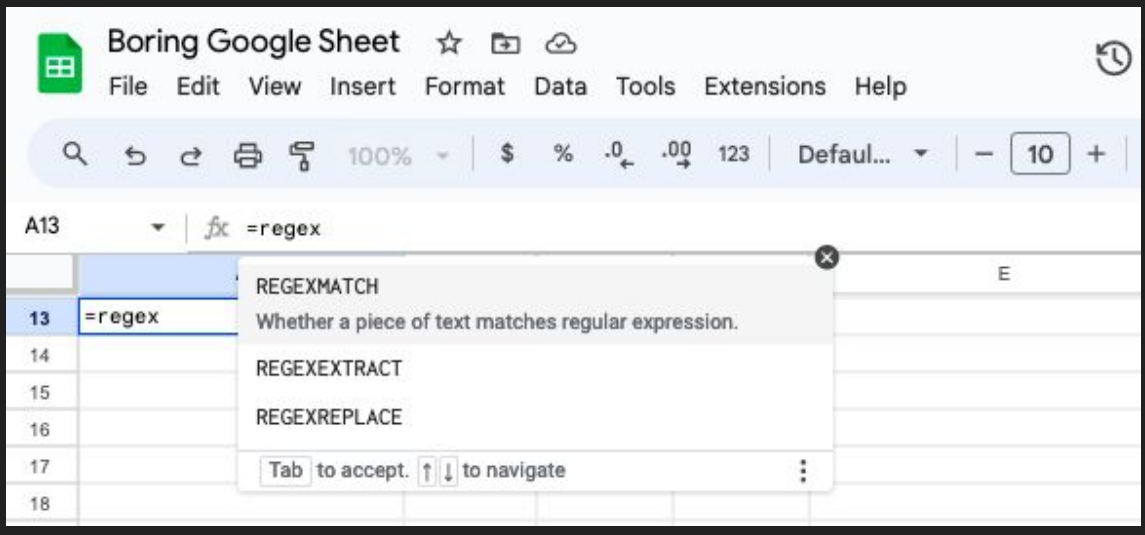
An important aspect of identity investigations (as well as other types) is the ability to search the data for a list of keywords. *bulk_extractor* provides the capability to do that through two different means. First, the ***find*** scanner is a simple regular expression finder that uses regular expressions. The ***find*** scanner looks through the data for anything listed in the global find list. The format of the find list should be rows of regular expressions while any line beginning with a # is considered a comment. The following is an excerpt from a sample find list file:

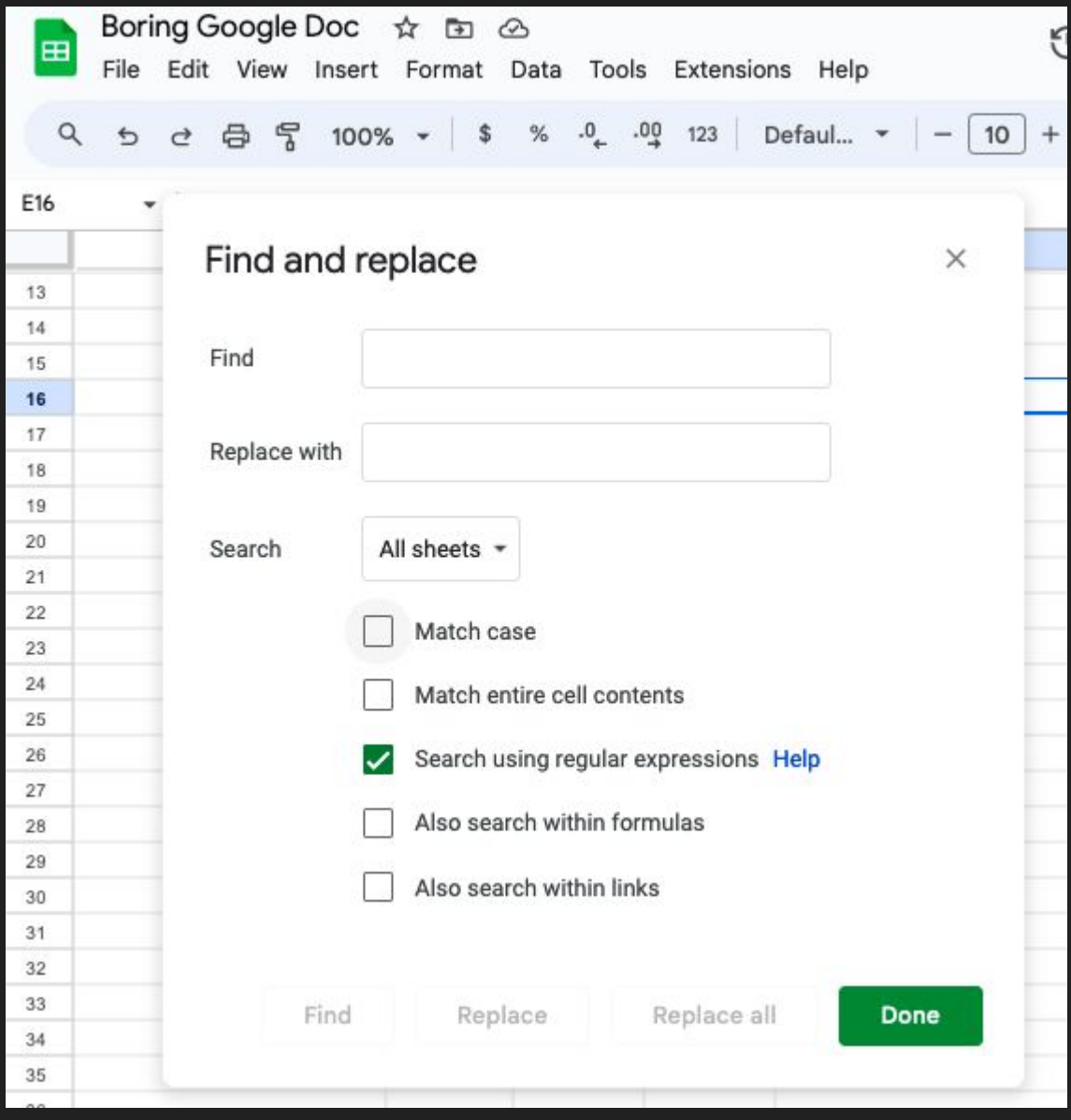
```
# This is a comment line
\b\d{1,3}\.\d{1,3}\.\d{1,3}\.\d{1,3}\b
# another comment line
/^[a-z0-9_-]{3,16}$/
```

The first regular expression from the above example, beginning with \b, looks for the following in order: a word boundary followed a digit repeated between 1-3 times, a digit repeated between 1-3 times, a digit repeated 1-3 times, a '.', a digit repeated 1-3 times,


OpenRefine allHazardNYYM
Permalink
Facet / Filter
Undo / Redo 1 / 1
118063 rows
Refresh
Reset all
Remove all
Show as: rows records
Show: 5 10 25 50 100 500 1000 rows
First name
invert reset
case sensitive
regular expression
All
ID
First name
Last name
Volume
Page
Note
1. 0 Enos Chase 1.1 4 Cornwal MM to m
2. 1 Joshua White 1.1 7 Coemans MM with w Deborah and 4 ch: Michael, Job, John & Phebe. See also pg 4 and C 733 Vol 3.7 pg 5
Mary Jacacks dt of Benjamin Jacacks, dec & Sarah. S of Tiddeman,dec & Elizabeth. See also C 733 Vol 1.1 pg 12
Nine-Partners MM, w of Samuel. See also C 733 Vol 3.7 pg 5
Jericho MM with w Elisabeth and
Replace
Find
case insensitive
whole word
regular expression
Leave blank to add the replacement string after each character.
Check "regular expression" to find special characters (new lines, tabulations…) or complex patterns.
Replace with
use \n for new lines, \t for tabulation, \\n for \n, \\t for \t.
If "regular expression" option is checked and finding pattern contains groups delimited with parentheses, $0 will return the complete string matching the pattern, and $1, $2… the 1st, 2nd… group.
OK
Cancel


OpenRefine allHazardNYYM
Permalink
Refresh
Custom text transform on column Note
Expression
Language
General Refine Expression Language (GREL)
value
.replace(/REGEX/, "string")
.split(/REGEX/
.find(/REGEX/
.contains(/REGEX/)
No syntax error.
Preview
History
Starred
Help
row
value
value .replace(/M\.?M\.?/, "Mo ...
1. Cornwal MM to m
Error:
java.lang.ArrayIndexOutOfBoundsExceptior
Index 0 out of bounds for length 0
2. Coemans MM with w Deborah and 4 ch: Michael, Job, John & Phebe. See also pg 4 and C 733 Vol 3.7 pg 5
false
3. Mary Jacacks dt of Benjamin Jacacks, dec & Sarah. S of Tiddeman,dec & Elizabeth.
true

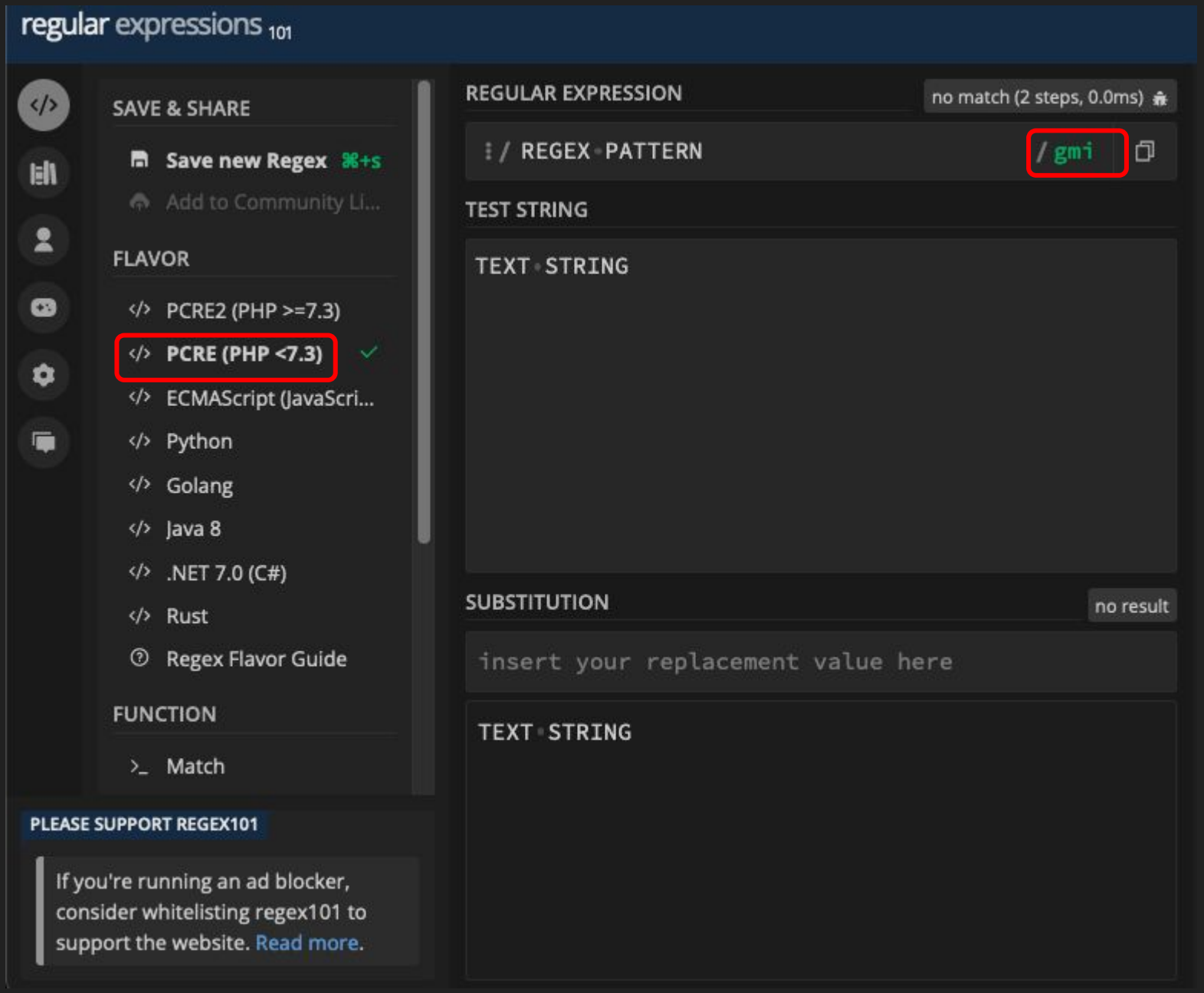

Boring Google Sheet
File Edit View Insert Format Data Tools Extensions Help
100% $ % .0 .00 123 Defaul... 10
A13 =regex
REGEXMATCH
Whether a piece of text matches regular expression.
REGEXEXTRACT
REGEXREPLACE
Tab to accept. to navigate
=regex


Boring Google Doc
File Edit View Insert Format Data Tools Extensions Help
100% $ % .0 .00 123 Defaul... 10
E16
Find and replace
Find
Replace with
Search
All sheets
Match case
Match entire cell contents
Search using regular expressions Help
Also search within formulas
Also search within links
Find
Replace
Replace all
Done

Learning the Syntax

regex101.com

bit.ly/bcc-regex



regex101.com

bit.ly/bcc-regex

Most Characters Match Themselves

Whitespace

Pattern	Meaning
\t	Tab
\n	Newline
\r	Carriage return

Boundaries

Pattern	Meaning
^	Start of line
$	End of line
\b	Word boundary

Character Classes

Pattern	Matches
[abcde]	catherine's 3 dogs
[^abcde]	catherine's 3 dogs
[3te]	catherine's 3 dogs
[^3te]	catherine's 3 dogs
[0-9]	catherine's 3 dogs
[^0-9]	catherine's 3 dogs
[a-e3]	catherine's 3 dogs

Character Class Shorthands

Shorthand	Equivalent to
.	[^\n\r]
\w	[a-zA-Z0-9_]
\W	[^\w]
\s	[\n\r \t]
\S	[^\s]
\d	[0-9]
\D	[^\d]
[^\w\s]	Punctuation
[\s\S]	Any character

Quantifiers

Pattern	Meaning
X?	0 or 1 instances of X
X*	0 or more instances of X
X+	1 or more instances of X

Quantifiers

Greedy	Lazy	Meaning
X?	X??	0 or 1 instances of X
X*	X*?	0 or more instances of X
X+	X+?	1 or more instances of X

(Capture Groups)

Pattern	Matches
ab+	ab, abb, abbb
(ab)+	ab, abab, ababab
(ab)\1	abab
(..)\1	abab, cdcd, adad
(\d\d)/\1	03/03, 07/07, 12/12

(Replacement Patterns)

\1, $1

Resources for learning more

- Tutorials
  - Library Carpentry: https://librarycarpentry.org/lc-data-intro/index.html
  - RegexOne: https://regexone.com/
  - The Programming Historian: https://programminghistorian.org/en/lessons/understanding-regular-expressions
  - Regular-Expressions.info: https://www.regular-expressions.info/tutorial.html
- Cheatsheets
  - Core RegEx: https://web.mit.edu/hackl/www/lab/turkshop/slides/regex-cheatsheet.pdf
  - Python's RegEx flavor: https://www.debuggex.com/cheatsheet/regex/python
- Tools
  - https://regexr.com/
  - https://regex101.com
- Exercises: http://alf.nu/RegexGolf