Technologies For Data Manipulation And Analysis

There's More than One Way to Groom a Cat(alog)

Presented by Annie Glerum and Kathryn Lybarger

OLAC Preconference October 26, 2017

Tools

Regular Expressions

What are regular expressions? (“regex” or “regexp”)

- Powerful search and replace
- Rather than just searching for specific things, you can search for kinds of things
- Symbols define a search pattern (similar to “wildcards”)
- This pattern may match your data
- Data may be modified or rearranged based on that matching

Examples:
Regex for finding data

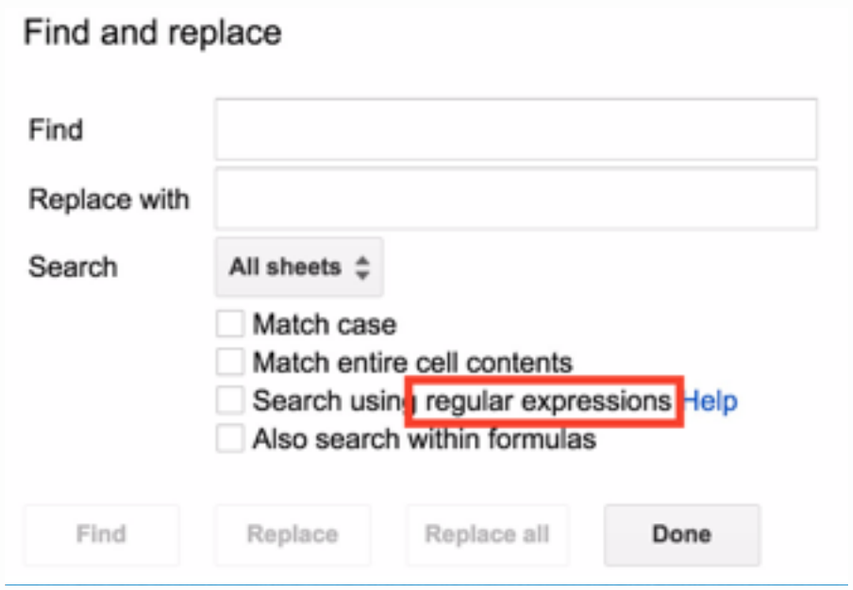
- Find all the lines that start with “2016”
- Find all 10- or 13-digit numbers
- Find all the fields like “Includes biblio … something something ... references.” (but not “bibliographical”)

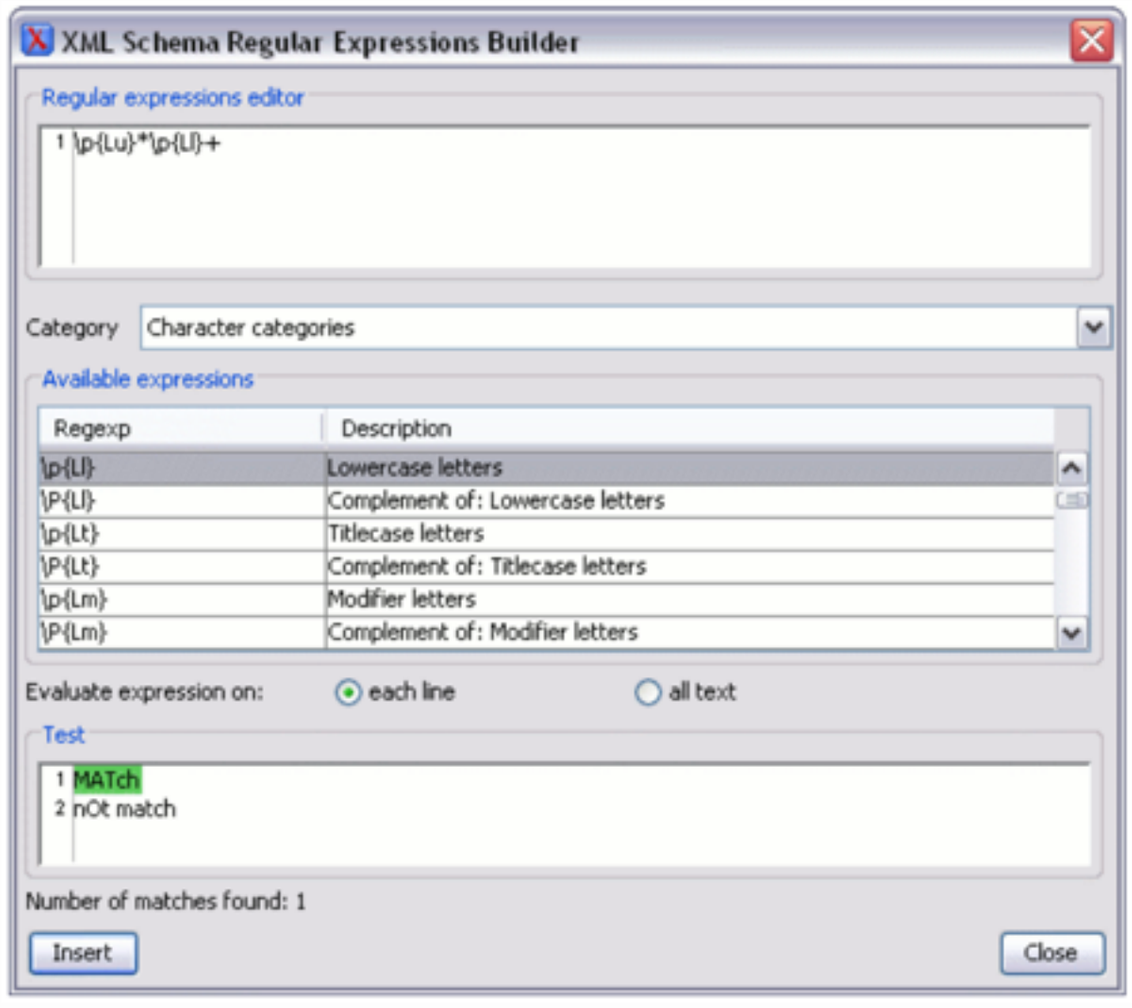
Examples:
Regex for modifying data

- Reformat all phone numbers from ###-###-#### to (###) ###-####
- Re-order names from LastName, FirstName -> FirstName LastName
  - Handling “Jr.” or “Sr.” correctly

Some common software in tech services has regex support

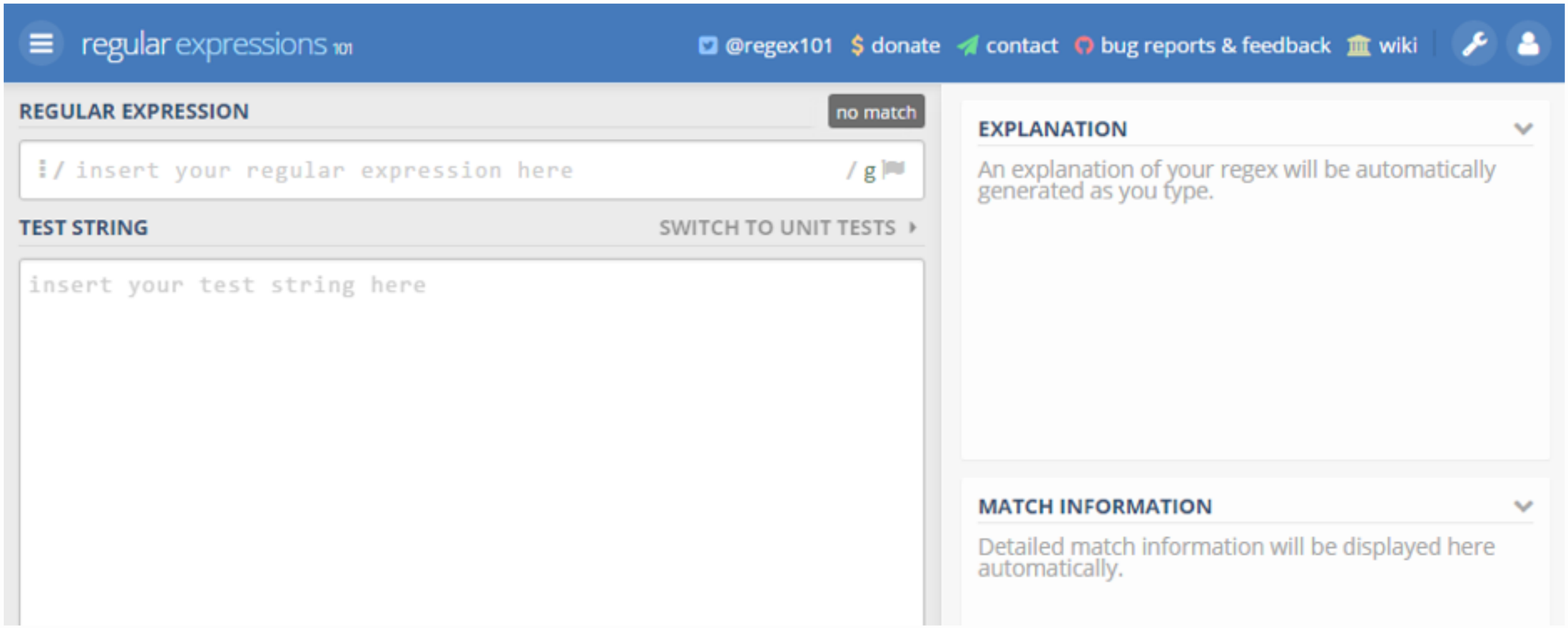
- MARCEdit
- Vendor-specific: Voyager's Global Data Change, etc.
- Google Sheets
- Text/XML Editors: vim, Sublime, EditPad, Oxygen
- Programming languages
- Databases like Oracle, MySQL
- Command line tools: grep, sed, awk






```
sed –e “s/now/meow/” file1 > file2
```

A nice learning tool: regex101.com



There are different “flavors” of regex

- Depends on software or language you’re using
  - e.g. Language with PCRE support, vim text editor
- Different features available
- Slightly different syntax to invoke some features
- BUT there is a common core
  - Once you know this, can look up details for your specific application

Forward slashes (/) are often used to indicate a regex

/ Regular Expression /

Forward slashes (/) are often used to indicate a regex

/ What does this pattern match? /

Literal characters:
the simplest regex!

/cat/

Matches:

cat
cataloging
s**cat**ter
wild**cat**

Literal characters:
the simplest regex!

/cat/

Does not match:

cart
CAT
act
c a t

Unless specified otherwise…

- Matching is case-sensitive
  - To match, the capitalization in your data must match the capitalization in your pattern
- Matching only part of the data is fine
  - It doesn’t have to be the whole word or line

BACK OF YOUR SHEET

/cat/

Regex Bingo!

- I will put a regex search pattern on the slide

- Search the indicated column (like B, I, N, G, O) – does the regex match any of those strings?

- If so, mark that space off!

- If you get five in a line, shout “BINGO!”

N

/FREE/

Meta-characters

\ ^ $ | ( ) . [ ] ? * + { }

- These are what give regular expressions their power.

- If the word or phrase you’re searching for has punctuation, it may not match as you intend

But what if I want to search for those?!

- You can “escape” such characters by preceding them with a backslash
- For example, if you wanted to search for a literal carat, include this in your regex: \^

Carat ^

Start of line anchor

/^dog/

Matches:

dog
dogwood
dog house
dogma

Carat ^

Start of line anchor

/^dog/

Does not match:

bulldog

What a cute dog!

dog

dooooooog

B

/^app/

/^rad/

Dollar sign $

End of line anchor

/ugh$/

Matches:

ugh

to**ugh**

do**ugh**

la**ugh**

Dollar sign $
End of line anchor

/ugh$/

Does not match:

taught

ugh, that's gross

UGH

upright

N

/one$/

G

/per$/

Vertical bar

This or that?

/ab|c/

Matches:

abdi**c**ate
cabaret
l**ab**oratory
Ja**c**k

Vertical bar

This or that?

/ab|c/

Does not match:

rad

blue

band

Canada

O

/ash|rr/

B

/DA|BB/

Dot (period):
It matches any character

/m.t.e/

Matches:

matte
mother
per**mitte**d
to**matoe**s

Dot (period):
It matches any character

/m.t.e/

Does not match:

smote
tempted
motormen
Mister

/t..ch/

N

/a.i.o/

Character classes
Square brackets – [ ]

- This part of the expression only matches one character
- Matches any character that appears in the brackets (but no others)

Character classes
Square brackets – [ ]

/m[ai]n/

Matches:

man
mine
wo**man**
swim**min**g

Character classes
Square brackets – [ ]

/m[ai]n/

Does not match:

main
chimney
women
am not

G

/s[oa]r/

O

/t[hr]o/

Character classes: some shorthand

- [a-z] instead of [abcdefghijklmnopqrstuvwxyz]
- [a-zA-Z] – any letter
- \d instead of [0123456789]
- \s – any whitespace like <space>, <tab>
- \w — any letter, number, or underscore

Character classes (excluding) - carat in square brackets [^]

- You can use similar notation to match any character EXCEPT what you specify
- This is the same carat character as used in the left anchor, but there should be no confusion

Character classes (excluding) - carat in square brackets [^]

/o[^ao]r/

Matches:

y**our**
aut**ogr**aph
c**obr**a
ourselves

Character classes (excluding) - carat in square brackets [^]

/o[^ao]r/

Does not match:

oar

poor

or

coder

B

/z[^ek]/

/Qu[^a]/

Quantifiers:
How many of a thing?

- These most recent patterns we've looked at have each matched exactly ONE character
  - But what if it is optional?
  - Or you can / must have more than one?
- These metacharacters directly follow patterns to specify such things

Question mark ? (zero or one time)

Thing before it is optional

/do?r/

Matches:

ar**dor**

dorsal

drone

An**dr**ew

Question mark ? (zero or one time)

Thing before it is optional

/do?r/

Does not match:

door

dOr

and

Lando

N

/ang?e/

G

/din?e/

Star (asterisk): (zero, one, or more!)
Thing before is optional, repeatable

/il*e/

Matches:
pumpkin **pie**
bibliophi**le**
M**ille**r
g**illle**ss

Star (asterisk): (zero, one, or more!)
Thing before is optional, repeatable

/il*e/

Does not match
fills
fire
dinner
tilted

O

/lan*c/

B

/smar*/

Plus sign: (one or more times)
Thing before is required, repeatable

/an+/

Matches:

an
annotate
m**ann**er
annnnnn

Plus sign: (one or more times)
Thing before is required, repeatable

/an+/

Does not match:
apple
minnow
Andover
nasty

/is+i/

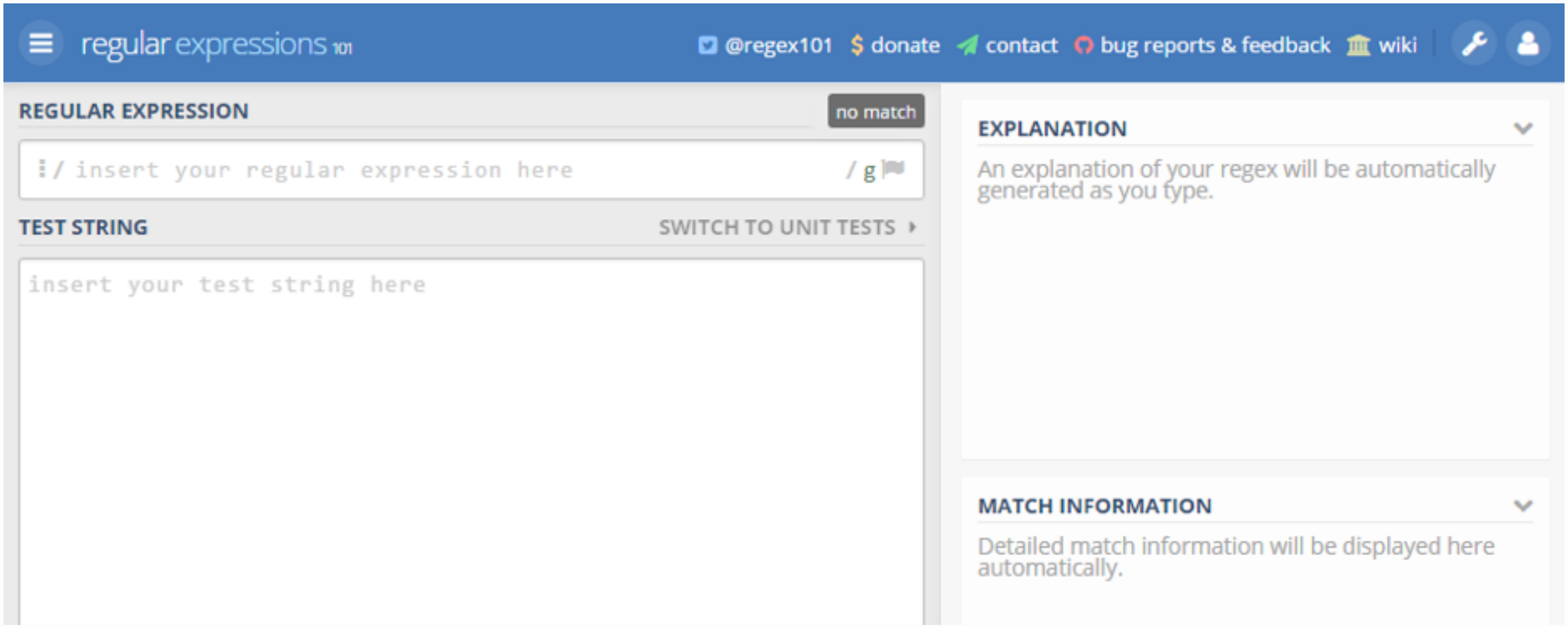
N

/ban+/

All of these can be used together

- /^(dog|raccoon) ?fo+d$/
- /[A-Z][a-z]* [A-Z][a-z]+/

Visit regex101.com



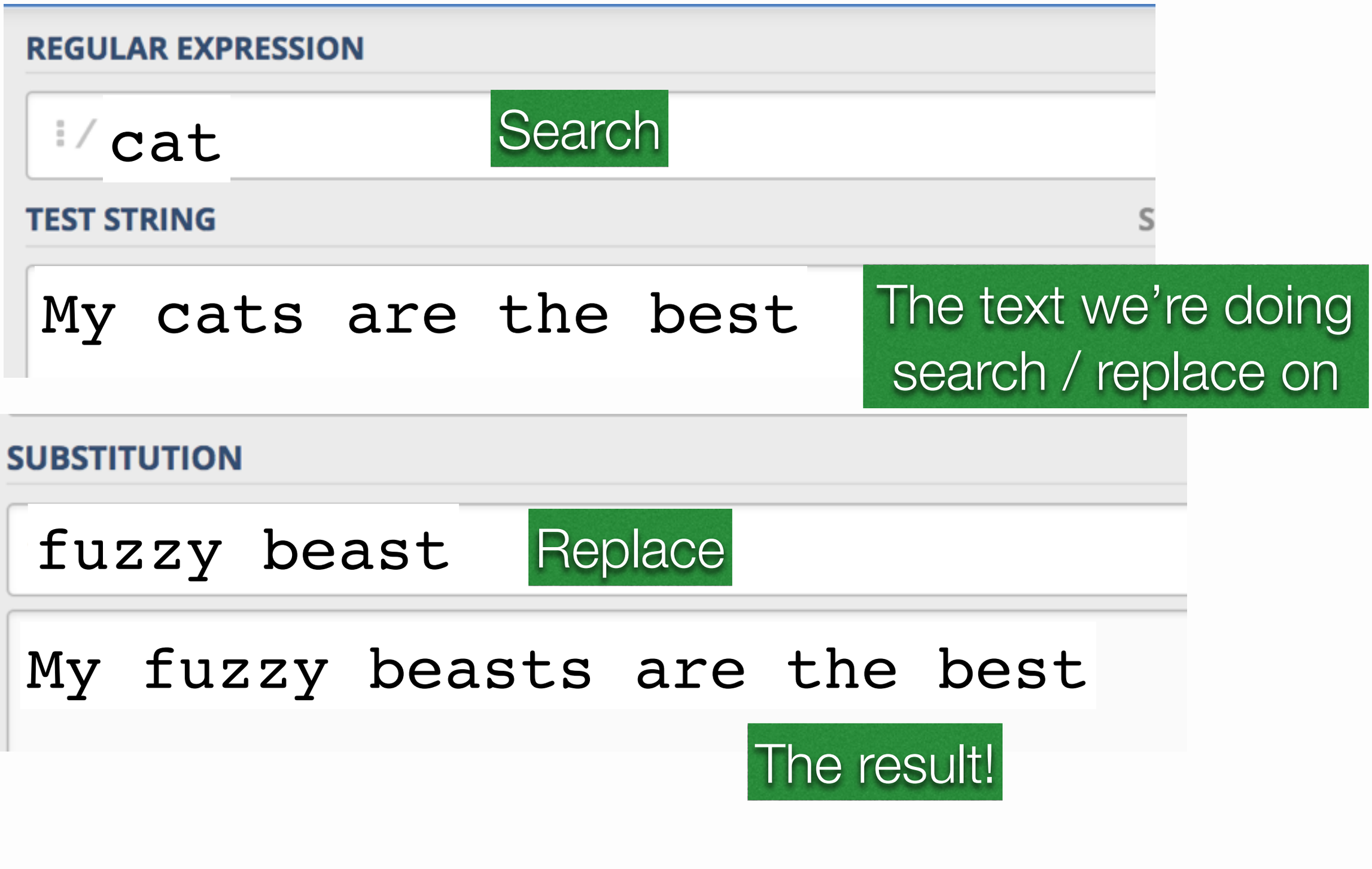
Going back to this example

/cat/

Matches:

cat
cataloging
s**cat**ter
wild**cat**

Search and replace



Search and replace

flexibility in searching

REGULAR EXPRESSION

```
/ cat|horse|dog
```

TEST STRING

```
My dogs are the best
```

SUBSTITUTION

```
fuzzy beast
```

```
My fuzzy beasts are the best
```

Parentheses ( )
Capturing patterns

- As you match patterns, you can “capture” parts of your data and rearrange them

- In the replacement pattern, refer to these parts by their “back references” like $1, $2, $3, … in order of parentheses in the pattern

Simple example: reformatting names

REGULAR EXPRESSION

/ ([A-Z]+) ([A-Z]+)

TEST STRING

KATHRYN LYBARGER

SUBSTITUTION

$2, $1

LYBARGER, KATHRYN

What does this match

```
/.*/
```

What does this match

/(.*)(.*)/

What does this do?

REGULAR EXPRESSION

```
(.*) (.*)
```

TEST STRING

```
JAMES EARL JONES
```

SUBSTITUTION

```
$2, $1
```

What is the result?

Regular expressions are “greedy”

- That first “capture group” (.*) has two choices:
  - Capture “James” and leave “Earl Jones” for the second
  - Capture “James Earl” and leave “Jones” for the second
- “Greedy” means it will capture as much as it can

Greedy capture

REGULAR EXPRESSION

```
(.*) (.*)
```

TEST STRING

```
JAMES EARL JONES
```

$1 = JAMES EARL

SUBSTITUTION

```
$2, $1
```

```
JONES, JAMES EARL
```

Think about this:
How to remove all GMD?

- Data: (lots of records like this)

  =100 1_ $a Sachar, Louis, $e author.

  =245 00 $a Holes $h [electronic resource] / $c Louis Sachar.

  =264 _1 $a New York : $b Random House, $c 2015.

- Goal:

- Remove 245 $h

References

- Regex101
  - http://www.regex101.com/
- Regular-Expressions.info
  - http://www.regular-expressions.info/
- Documentation for your particular software / application
  - Where to use regex
  - Which features are available
  - How to invoke them