Regular Expressions

There's More than One Way to Groom a Cat(alog): Technologies for Data Analysis and Manipulation. OLAC Preconference October 26, 2017

Common meta-characters for matching

Literal characters	/sew/ matches **sew** and hou**sew**ork
Start of line anchor **^**	/^cat/ matches **cat** and **cat**aloging
End of line anchor **$**	/ran$/ matches **ran** and catama**ran**
This or that? **|**	/dd|p/ matches a**dd**er and **p**ig
Dot (almost anything) **.**	/d.n.r/ matches **diner** and **donor**
Character class **[ ]** - can use ranges like [a-z] [A-Z] [0-9]	/a[rl]e/ matches **are** and t**ale**s
Character class excluding **[^]**	/a[^rl]e/ matches t**ake**n and r**ate**
Optional **?**	/sh?y/ matches **shy** and bu**sy**
Repeated (optional) *****	/b*y/ matches wh**y**, ba**by**, ho**bby**
Repeated (required) **+**	/un+/ matches **un**der and f**unn**y
Repeated specified number of times { }	/at{2}/ matches m**att**er and fl**att**en

If you actually want to match one of these metacharacters, escape it by preceding with a backslash, like \+ or \$

Multiple meta-characters can be used within a single expression, for example:

- /^tiger$/ matches **tiger** (but nothing else)
- /.*/ matches everything
- /^$/ matches only blank lines
- /h.*e/ matches **he**, **home**, **housemate**
- /a[rg]+e/ matches t**arge**t, **agre**e, w**arre**n, p**age**

Shorthand

These notations can be used in regular expressions, inside or outside character classes:

- \w – "word characters", matches any letter, number, or underscore
- \d – digits, matches any digit 0 through 9
- \s – "whitespace", matches any whitespace character, like <space> or <tab>

Flags

You can make regular expressions behave differently by using flags; depending on the software you're using, this may be set in various ways, such as a command line switch, a checkbox, or letters added to the end of the regex. Some common flags include:

- Case sensitivity – does it matter if letters are uppercase or lowercase?
- Global – Should your search/replace only find one match, or as many as it can?

Capturing with ( )

“Capture” parts of the text with parentheses, refer to captured parts with numbers. By default, regular expressions are “greedy” and will capture as much as they can.

Applying this expression…	**to this text…**	**captures this:**
/(Bob|Robert) Stark/	Robert Stark	$1 = Robert
/(.*) (.*)/	Stanley Yelnats	$1 = Stanley $2 = Yelnats
/(.*) (.*)/	Tommy Lee Jones	$1 = Tommy Lee $2 = Jones
/([A-Z]+ ([A-Z]+))/	STOP SIGN	$1 = STOP SIGN $2 = SIGN

Substitution

You can do search and replace with regular expressions, using components that you've captured in the "replace" string:

Search regex…	**In this string…**	**Replace with…**	**Result**
/^(.*)$/	Sparky	Hello, $1	Hello, Sparky
/^(.*)$/	Magic	-=$1=-	-=Magic=-
/([A-Z]+).*/	ABRAcadabra1	$1	ABRA
/(.*) (.*)/	Martha Jones	$2, $1	Jones, Martha
/(.*) (.*)/	Ruth Bader Ginsburg	$2, $1	Ginsburg, Ruth Bader

Further study and practice

Regular-Expressions.info   http://www.regular-expressions.info/

> Extensive reference site with a tutorial, clear description of regex features, and documentation of their support in various software and programming languages

Regular Expressions 101 https://regex101.com/

> An interactive sandbox for experimenting with regular expressions. Supports common flavors, provides clear immediate feedback on what expressions match, supported by embedded quick reference.

Software documentation

> Are you using software that supports regular expressions? Check that! They will often specify what flavor of regex they generally support, as well as any additional features. Many provide examples and tutorials as well!