

UNIVERSITY *of* WASHINGTON

Data Science UW

Methods for Data

Analysis

Regex Tutorial
Nick McClure



WHENEVER I LEARN A
NEW SKILL I CONCOCT
ELABORATE FANTASY
SCENARIOS WHERE IT
LETS ME SAVE THE DAY.

OH NO! THE KILLER
MUST HAVE FOLLOWED
HER ON VACATION!



BUT TO FIND THEM WE'D HAVE TO SEARCH
THROUGH 200 MB OF EMAILS LOOKING FOR
SOMETHING FORMATTED LIKE AN ADDRESS!



IT'S HOPELESS!

EVERYBODY STAND BACK.



I KNOW REGULAR
EXPRESSIONS.



W

Regular Expressions (Regex)

- > Regular Expressions: a sequence of characters that define a search pattern for string matching.
- > Example:
 - Find all numbers in the string “123 Main St. was built in the 50’s.”
 - > Should return “123” and “50”.
 - Find all 3 digit numbers:
 - > Should only return “123”.
 - We want to match the phrases:
 - > “The color is grey”, “The colour is grey”, “The color is gray”, or “The colour is grey”.
 - > ‘|’ (bar) = or operator and ‘?’ = optional operator.
 - > Regexes:
 - **“The colour?r is gr(a|e)y”**
 - **“The (colour|color) is (gray|grey)”**
 - > Both will match.



Regex Literals and Special Characters

> Literal matches:

- **ABCDEF...**
- **abcdef...**
- **1234567890**
- **!,@#%&**
- A space: ' '
- The above lines will be matched literally.

> Special Characters:

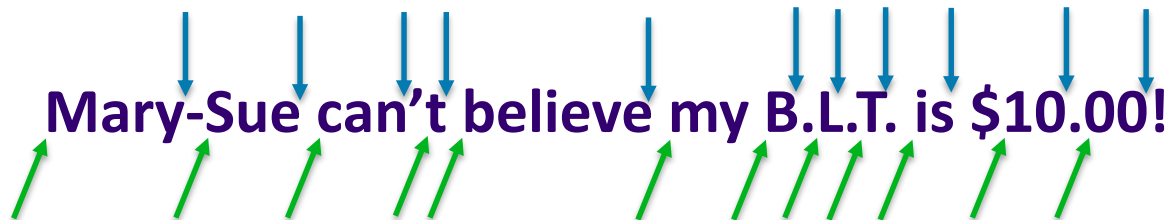
- **|()[]{}^\$.?*\+**
- To match the above characters literally, they must be escaped with a backslash.
- **'\\$'** will look for a literal dollar sign.
- **'\\'** will look for a backslash.



Regex Special Characters

- > Sometimes we want to search for text and where it is.
- > ^ = start of a line
- > \$ = end of a line
- > \b = word boundary
- > \B = not a word boundary
- > Examples:
 - ‘**^Subject**’: Find a line that starts with ‘Subject’
 - ‘**^\$**’: Find an empty line.
 - ‘**^**’: Find any line. Effectively meaningless, all lines have beginnings, even empty lines.
- > **Start** and **end** boundaries:

Mary-Sue can't believe my B.L.T. is \$10.00!

A diagram illustrating word boundaries in the sentence "Mary-Sue can't believe my B.L.T. is \$10.00!". Blue arrows point down to the start of each word: "Mary", "Sue", "can't", "believe", "my", "B.L.T.", "is", "\$10.00!". Green arrows point up to the end of each word: "Mary", "Sue", "can't", "believe", "my", "B.L.T.", "is", "\$10.00!".

W

Regex Character Classes

- > “[]” denotes a character class.
 - **[Math]** will match a single character that is either a M, a, t, or h
 - **[a-z]** will match any lower case letter a through z
 - **[A-Za-z0-9]** will match any upper case, lower case, or number
 - **[^Math]** will match any character that is not M, a, t, or h
 - Note that some characters behave differently inside a character class than outside of it.
 - **‘mat[^h]’** matches matt but not math.
 - > Note: It may or may not match ‘mat’. The search tool you use may remove the end of line character (in case it will not match). And if there is an end of line character, it will match.
- Also, the placement of the ‘not’ character in the class matters.
- **[Ma^th]** will match any of the characters M, a, t, h, or ^

W

Regex Character Classes (Shorthand)

- > **\d** = [0-9]
- > **\w** = [0-9a-zA-Z_]
- > **\s** = [\t(?:\n|\r\n)] (whitespace)
- > **\D** = [^0-9]
- > **\W** = [^0-9a-zA-Z_]
- > **\S** = [^\t(?:\n|\r\n)] (not whitespace)

W

Regex Repetitions and Optional Items

- > **+** Matches the prior object at least once. (one or more).
- > ***** matches any number, including none of the prior object.
- > **?** Is placed after an optional item.
- > Matching the month July can be written:
 - **(July|Jul)**
 - **July?**
- > Or for the fourth of July:
 - **July? 4(th)?**

	Min Required	Max to Try	Meaning
?	0	1	One allowed, none required
*	0	No Limit	Unlimited allowed, non required
+	1	No Limit	Unlimited allowed, one required



Regex Exact Repitions

- > We can specify exactly how many repetitions we want with {}
- > **[0-9]{8}** Matches an 8-digit number
- > **[0-9]{8,}** Matches an 8 or more digit number
- > **[0-9]{2,5}** Matches a 2 digit number, 3 digit number, a 4 digit number or a 5 digit number.



Regex Examples

> A string within double quotes

– “[^”]*”

> Dollar Amount

– \#[0-9]+(\.[0-9][0-9])?

> An HTTP URL

– http://[-a-z0-9_.:]+/[-a-z0-9_:@&?+=,./~*%\$]*\.(com|html?|edu)

> Phone Number

– \(\d{3}\)?[\s-]\d{3}[\s-]\d{4}



Regex Back Referencing

- > **()** is considered a grouping. We can reference these groupings (like we did with the **?**).
- > Escaping a number references the groups in the regular expression.
- > **([A-W])([A-Z])=\1\2** Here, the '**\1**' references the first group and the **\2** references the second group.
 - This regex will match AZ=AZ and WY=WY, but not AZ=WY.

W

Regex Look Aheads and Look Behinds

- > **(?=regex1)regex2** is a look ahead. The regex2 only matches in places where regex1 will match ahead.
- > **(?<=regex1)regex2** is a look behind. The regex2 only matches after the regex1 matches.
- > **(?=Tomato)Tom** matches the Tom part of Tomato, but not Tomas.
- > **(?<=T)here** matches the here part of There, but not just here

W

Regex POSIX Expressions

- > **[alnum:]** – alpha numeric characters = **[A-Za-z0-9]**
- > **[alpha:]** – Alphabetic characters = **[A-Za-z]**
- > **[blank:]** – spaces and tabs
- > **[digit:]** – numbers = **[0-9]**
- > **[lower:]** – lowercase alphabetic characters = **[a-z]**
- > **[upper:]** – uppercase alphabetic characters = **[A-Z]**
- > **[punct:]** - **!"#\$%&'()*+,-._/:\;@^`~{|}**
- > **[space:]** – space characters = space, tab, newline, vertical tab, form feed, carriage return

- > <https://regex101.com>
- > <http://regexpal.com>

- > Regex in R tutorial

- > A great book with LOTS of detail: “Mastering Regular Expressions” by Jeffrey Friedl. (O'Reilly Series)

