MAKE PARSING GREAT AGAIN

(with regular expressions)

Robert Gebeloff @gebeloffnyt
The New York Times
Dataharvest 2017

So much of what we do in data journalism is fun. Writing queries. Making maps. Holding powerful people accountable, etc.

What's not to love? Cleaning and parsing data, for one. There's usually a time in every project where you're excited to dig in, only to realize that the data you want is not in easily diggable format.

This tutorial will introduce you to regular expression, a powerful syntax library that can help you clean-up your data, or in this case, turn text into something tabular that you can pop into your spreadsheet or database program.

Regular expressions, or RegEx for short, exists in numerous flavors in a wide variety of scripting environments. These different variations might carry slightly different syntax conventions, but the concepts are the same.

For this session, we're going to use a Web site that is designed to help you write Regex code and extract your results. For dirty data, we're going to turn to none other than Donald J. Trump, and his Twitter feed.

But before we begin, let me go over some important concepts.

RegEx is all about patterns. The simplest way to think of it is to compare it to a traditional search operation. In a text file, you can search for a specific string, like "Belgium", or in SQL, you can use a wildcard character and search for countries that begin with "B*".

But in RegEx, you can create very sophisticated search logic. Find any four-letter word that begins with T:

```
(T[a-z]{3} )
```

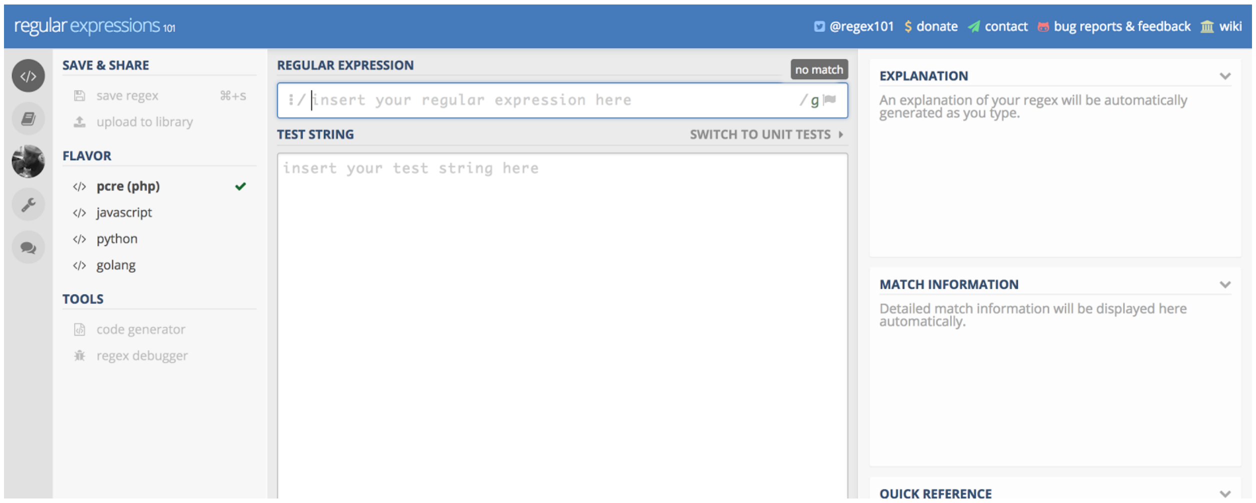
That expression is interpreted as "Starts with a capital 'T', and is followed by three lower-case letters and then a space." It will match "This" "Them" "That" but not this, these or those.

What's difficult for beginners is learning all of the possible RegEx tools at your disposal. In some lessons, we'd dive in with a tour of all of the common tools, but the philosophy here is that it's better to learn while working on an actual job.

So hello @realDonaldTrump

In this exercise, we're going to take 499 Trump tweets from the first quarter of 2017 and convert them into a tab-delimited format that we can save into a spreadsheet. We want to capture the dates and time of tweets, the contents of the tweets, and the software he tweeted with, which is listed at the end of each tweet. The data comes from the Trump Twitter Archive (see http://www.trumptwitterarchive.com/about for more info).

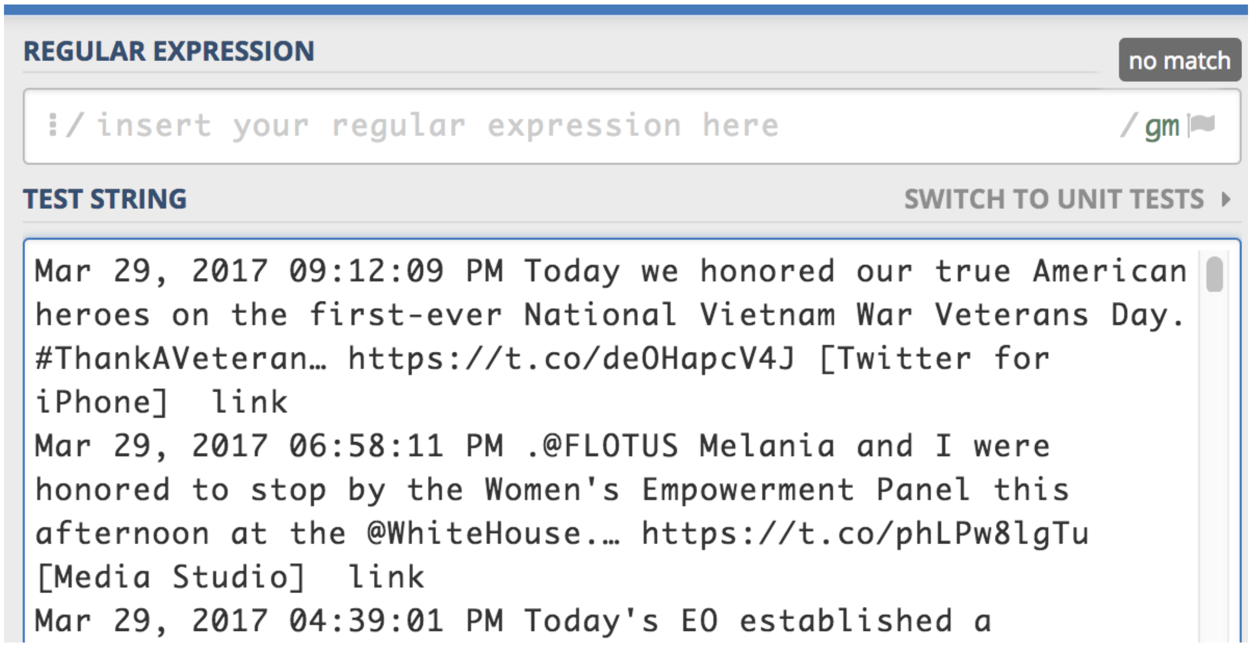
We are going to do our parsing using a great online tool, http://regex101.com, and I've already created a the Trump tweet file here (http://bit.ly/dhtrumptweets). So open each site in separate browser windows.



Regex101 is a fabulous resource because it allows you to create your Regex code online and then bring it back into whatever environment you're working in. For example, if you're writing a program in Python that loops through thousands of documents and extracts certain information with Regex, you can test your code here before running it on the big batch.

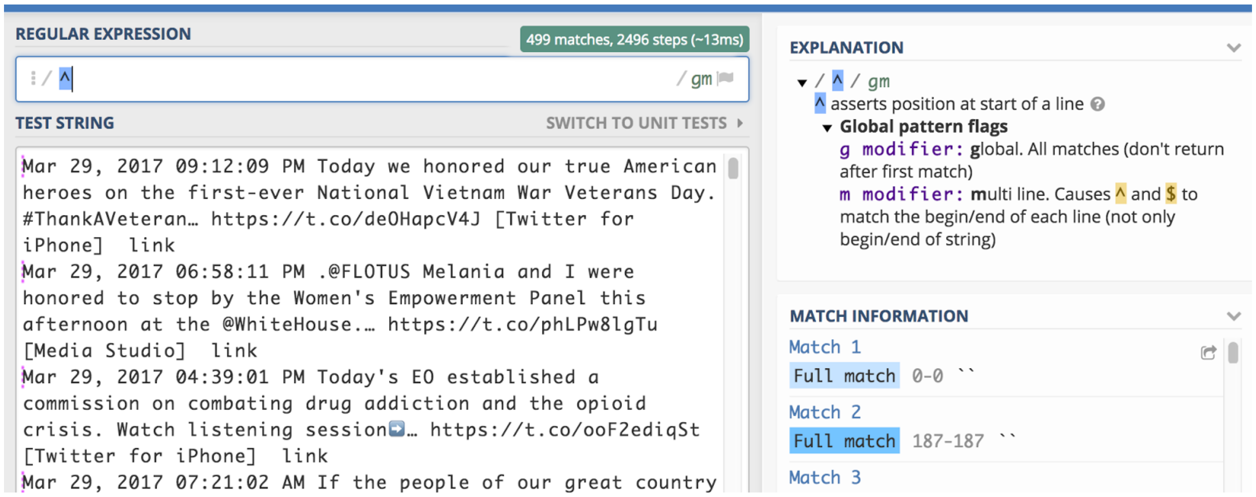
In our case, we're going to parse the Trump tweets and then copy and paste the results into a spreadsheet.

The first step you want to do is get your data into the tool, so copy and paste the Trump data into the big box labeled "Test String". Also at this time, let the parser know that each Tweet in this dataset spans multiple lines by clicking on the "g" in the Regular Expression box and adding the "m".



```
Mar 29, 2017 09:12:09 PM Today we honored our true American
heroes on the first-ever National Vietnam War Veterans Day.
#ThankAVeteran… https://t.co/deOHapcV4J [Twitter for
iPhone]  link
Mar 29, 2017 06:58:11 PM .@FLOTUS Melania and I were
honored to stop by the Women's Empowerment Panel this
afternoon at the @WhiteHouse.… https://t.co/phLPw8lgTu
[Media Studio]  link
Mar 29, 2017 04:39:01 PM Today's EO established a
```

To get started, we need to enter the symbol that tells Regex to start our quest at the beginning of each new line. That is done with the symbol ^


REGULAR EXPRESSION

499 matches, 2496 steps (~13ms)

```
/ ^ / gm
```

TEST STRING

SWITCH TO UNIT TESTS

```
Mar 29, 2017 09:12:09 PM Today we honored our true American
heroes on the first-ever National Vietnam War Veterans Day.
#ThankAVeteran… https://t.co/deOHapcV4J [Twitter for
iPhone]  link
Mar 29, 2017 06:58:11 PM .@FLOTUS Melania and I were
honored to stop by the Women's Empowerment Panel this
afternoon at the @WhiteHouse.… https://t.co/phLPw8lgTu
[Media Studio]  link
Mar 29, 2017 04:39:01 PM Today's EO established a
commission on combating drug addiction and the opioid
crisis. Watch listening session➡… https://t.co/ooF2ediqSt
[Twitter for iPhone]  link
Mar 29, 2017 07:21:02 AM If the people of our great country
```

EXPLANATION

/ ^ / gm
^ asserts position at start of a line
Global pattern flags
g modifier: global. All matches (don't return after first match)
m modifier: multi line. Causes ^ and $ to match the begin/end of each line (not only begin/end of string)

MATCH INFORMATION

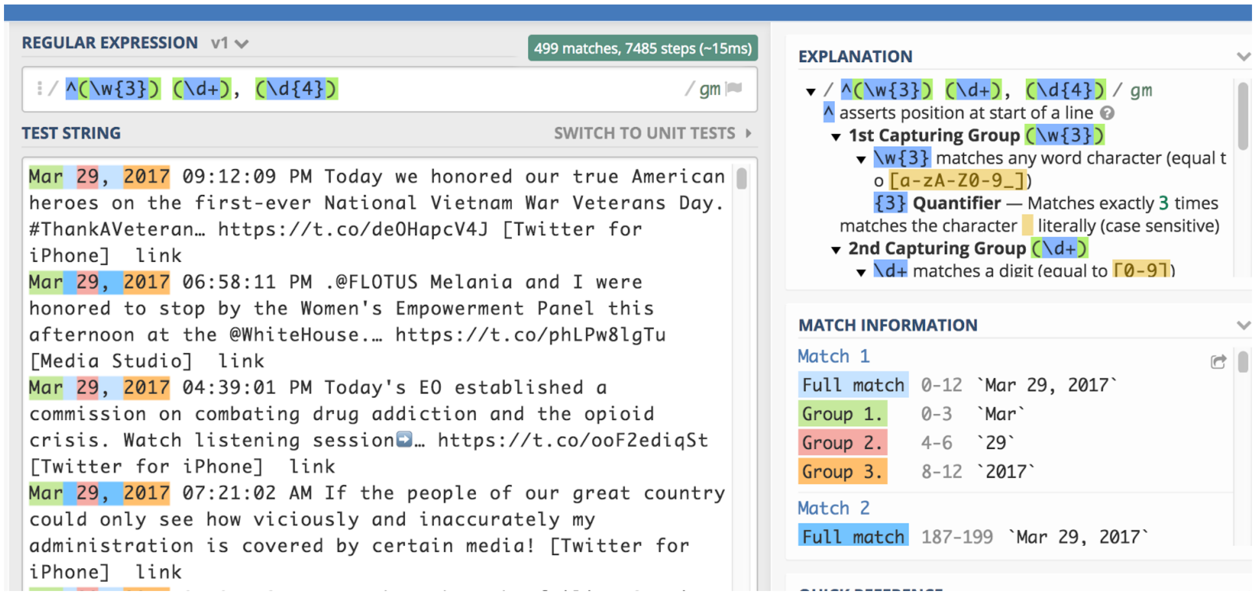
Match 1
Full match 0-0 ``
Match 2
Full match 187-187 ``
Match 3

A few things to note:
-- The beginning of each line in the Test String is now marked by dots
-- A box appears above our Regular Expression letting us know there are 499 matches
-- An explanation appears on the right explaining to us in plain language what our expression is doing
-- More information about our match appears in the "Match Information" box.
-- Below that, and not pictured, is a handy Regex cheat sheet to guide your way.

Ok, so let's capture the month. If you browse the text, you'll see that Twitter uses a three-letter abbreviation for a month symbol. So we are going to create our first capture "group" (a future column in our database) and tell it to grab a three-letter string. It will look like this: (\w{3}) The parens are used to declare the group, and the expression inside literally means find a letter or number \w that repeats three times {3}. Notice how all the information in the regex101 updates to show you how the command is playing out. The \ precedes many of the regex commands.

Next we need to capture the day of the month, which is always numeric but can be one or more digits. So first enter a space - we need to account for the space between the month and day, and then enter (\d+). As you might guess, \d means a digit, but in this case, the + sign means one or more of.

Ok, next up is the year. Can you guess how we'd capture that? Try it yourself before turning to the next page…



REGULAR EXPRESSION v1

499 matches, 7485 steps (~15ms)

```
/ ^(\w{3}) (\d+), (\d{4}) / gm
```

TEST STRING — SWITCH TO UNIT TESTS

```
Mar 29, 2017 09:12:09 PM Today we honored our true American
heroes on the first-ever National Vietnam War Veterans Day.
#ThankAVeteran… https://t.co/deOHapcV4J [Twitter for
iPhone]  link
Mar 29, 2017 06:58:11 PM .@FLOTUS Melania and I were
honored to stop by the Women's Empowerment Panel this
afternoon at the @WhiteHouse.… https://t.co/phLPw8lgTu
[Media Studio]  link
Mar 29, 2017 04:39:01 PM Today's EO established a
commission on combating drug addiction and the opioid
crisis. Watch listening session➡… https://t.co/ooF2ediqSt
[Twitter for iPhone]  link
Mar 29, 2017 07:21:02 AM If the people of our great country
could only see how viciously and inaccurately my
administration is covered by certain media! [Twitter for
iPhone]  link
```

EXPLANATION

/ ^(\w{3}) (\d+), (\d{4}) / gm
^ asserts position at start of a line
1st Capturing Group (\w{3})
\w{3} matches any word character (equal to [a-zA-Z0-9_])
{3} Quantifier — Matches exactly 3 times
matches the character   literally (case sensitive)
2nd Capturing Group (\d+)
\d+ matches a digit (equal to [0-9])

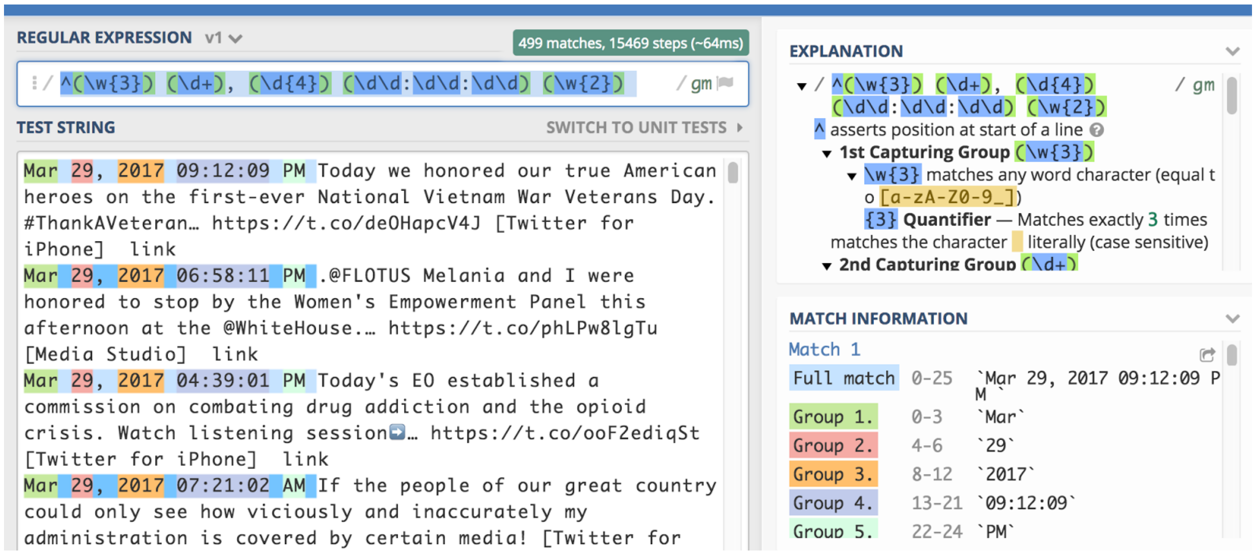
MATCH INFORMATION

Match 1		
Full match	0-12	`Mar 29, 2017`
Group 1.	0-3	`Mar`
Group 2.	4-6	`29`
Group 3.	8-12	`2017`
Match 2		
Full match	187-199	`Mar 29, 2017`

Yes, combining the techniques from group 1 and group 2, we tell Regex to expect a comma and a space, and to then create a new group with four-digit string.

Note that in data work, dates and times formats can be messy, A valid date in Excel might not be a valid format in MySQL. Therefore, it's often wise to initially work with dates as components -- month, day, year -- as opposed to trying to capture the entire date in the source data's format. For the time, we're going to accept the source format, but keep the AM/PM flag as a separate column.

```
^(\w{3}) (\d+), (\d{4}) (\d\d:\d\d:\d\d) (\w{2})
```



REGULAR EXPRESSION v1

499 matches, 15469 steps (~64ms)

```
/ ^(\w{3}) (\d+), (\d{4}) (\d\d:\d\d:\d\d) (\w{2}) / gm
```

TEST STRING — SWITCH TO UNIT TESTS

```
Mar 29, 2017 09:12:09 PM Today we honored our true American
heroes on the first-ever National Vietnam War Veterans Day.
#ThankAVeteran… https://t.co/deOHapcV4J [Twitter for
iPhone]  link
Mar 29, 2017 06:58:11 PM .@FLOTUS Melania and I were
honored to stop by the Women's Empowerment Panel this
afternoon at the @WhiteHouse.… https://t.co/phLPw8lgTu
[Media Studio]  link
Mar 29, 2017 04:39:01 PM Today's EO established a
commission on combating drug addiction and the opioid
crisis. Watch listening session➡… https://t.co/ooF2ediqSt
[Twitter for iPhone]  link
Mar 29, 2017 07:21:02 AM If the people of our great country
could only see how viciously and inaccurately my
administration is covered by certain media! [Twitter for
```

EXPLANATION

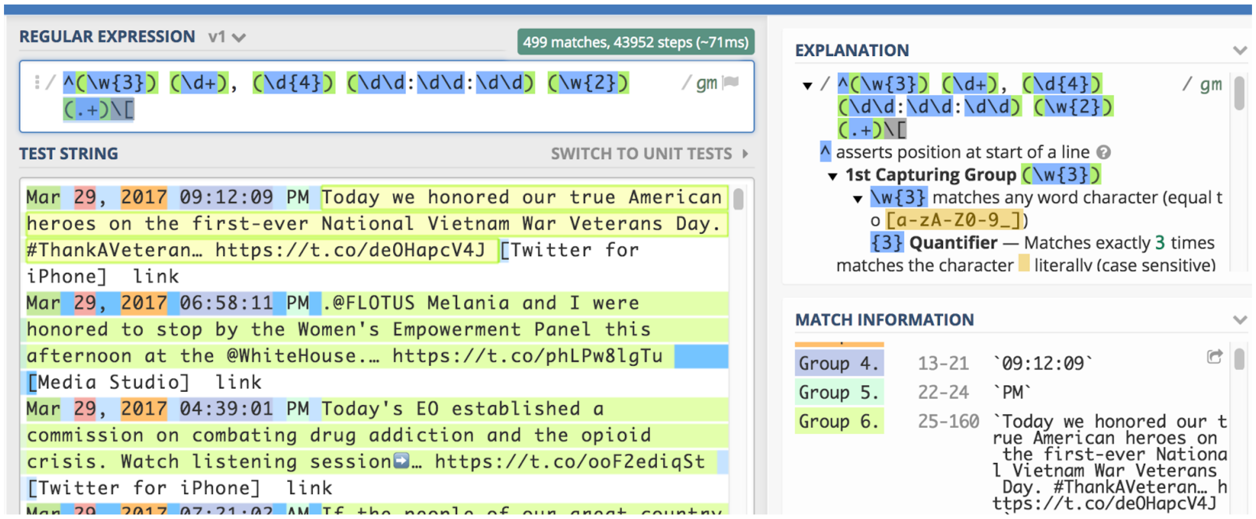
/ ^(\w{3}) (\d+), (\d{4}) (\d\d:\d\d:\d\d) (\w{2}) / gm
^ asserts position at start of a line
1st Capturing Group (\w{3})
\w{3} matches any word character (equal to [a-zA-Z0-9_])
{3} Quantifier — Matches exactly 3 times
matches the character   literally (case sensitive)
2nd Capturing Group (\d+)

MATCH INFORMATION

Match 1		
Full match	0-25	`Mar 29, 2017 09:12:09 PM`
Group 1.	0-3	`Mar`
Group 2.	4-6	`29`
Group 3.	8-12	`2017`
Group 4.	13-21	`09:12:09`
Group 5.	22-24	`PM`

Ok, so now we're up to the Tweet itself. In this case, there is no pattern we want to match. Instead, we want to capture everything that begins one space after the AM/PM flag up to the "[" character. We also have to tell Regex that we literally mean a "[" character, since [could also be party of the regex language.

We can do it this way: ` (.+)\[` This literally means, after a space, create a group that starts with any character and goes on indefinitely. But the group should end when it hits the first "[" sign, which in this case is preceded by a \ to signify that we mean the literal "[", not the regex [syntax.


REGULAR EXPRESSION v1

499 matches, 43952 steps (~71ms)

```
/ ^(\w{3}) (\d+), (\d{4}) (\d\d:\d\d:\d\d) (\w{2}) (.+)\[ / gm
```

TEST STRING

SWITCH TO UNIT TESTS

```
Mar 29, 2017 09:12:09 PM Today we honored our true American
heroes on the first-ever National Vietnam War Veterans Day.
#ThankAVeteran… https://t.co/deOHapcV4J [Twitter for
iPhone]  link
Mar 29, 2017 06:58:11 PM .@FLOTUS Melania and I were
honored to stop by the Women's Empowerment Panel this
afternoon at @WhiteHouse.… https://t.co/phLPw8lgTu
[Media Studio]  link
Mar 29, 2017 04:39:01 PM Today's EO established a
commission on combating drug addiction and the opioid
crisis. Watch listening session➡… https://t.co/ooF2ediqSt
[Twitter for iPhone]  link
```

EXPLANATION

```
/ ^(\w{3}) (\d+), (\d{4}) (\d\d:\d\d:\d\d) (\w{2}) (.+)\[ / gm
```

^ asserts position at start of a line

1st Capturing Group (\w{3})

\w{3} matches any word character (equal to [a-zA-Z0-9_])

{3} **Quantifier** — Matches exactly 3 times

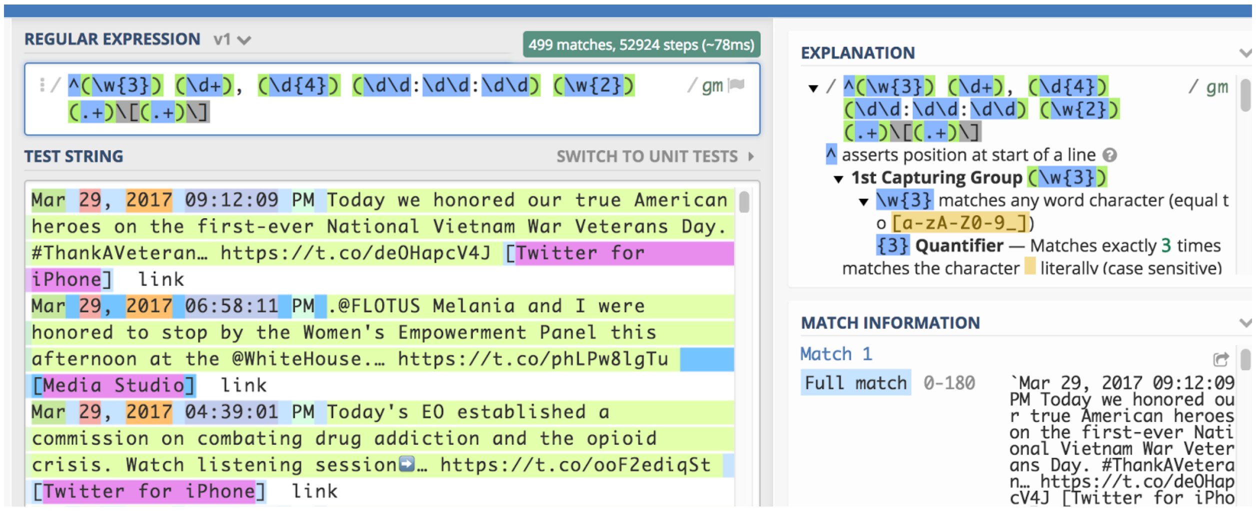
matches the character literally (case sensitive)

MATCH INFORMATION

Group 4.	13-21	`09:12:09`
Group 5.	22-24	`PM`
Group 6.	25-160	`Today we honored our true American heroes on the first-ever National Vietnam War Veterans Day. #ThankAVeteran… https://t.co/deOHapcV4J

The last part -- "Twitter for iPhone", "Media Studio" -- we can capture with similar syntax.

Give it a try before turning the page to see the results….


REGULAR EXPRESSION v1

499 matches, 52924 steps (~78ms)

```
/ ^(\w{3}) (\d+), (\d{4}) (\d\d:\d\d:\d\d) (\w{2}) (.+)\[(.+)\]    / gm
```

TEST STRING   SWITCH TO UNIT TESTS

```
Mar 29, 2017 09:12:09 PM Today we honored our true American
heroes on the first-ever National Vietnam War Veterans Day.
#ThankAVeteran… https://t.co/de0HapcV4J [Twitter for
iPhone]  link
Mar 29, 2017 06:58:11 PM .@FLOTUS Melania and I were
honored to stop by the Women's Empowerment Panel this
afternoon at the @WhiteHouse.… https://t.co/phLPw8lgTu
[Media Studio]  link
Mar 29, 2017 04:39:01 PM Today's EO established a
commission on combating drug addiction and the opioid
crisis. Watch listening session➡… https://t.co/ooF2ediqSt
[Twitter for iPhone]  link
```

EXPLANATION

```
/ ^(\w{3}) (\d+), (\d{4}) (\d\d:\d\d:\d\d) (\w{2}) (.+)\[(.+)\]  / gm
^ asserts position at start of a line
1st Capturing Group (\w{3})
\w{3} matches any word character (equal t
o [a-zA-Z0-9_])
{3} Quantifier — Matches exactly 3 times
matches the character   literally (case sensitive)
```

MATCH INFORMATION

```
Match 1
Full match  0-180  `Mar 29, 2017 09:12:09
                   PM Today we honored ou
                   r true American heroes
                   on the first-ever Nati
                   onal Vietnam War Veter
                   ans Day. #ThankAVetera
                   n… https://t.co/de0Hap
                   cV4J [Twitter for iPho
```

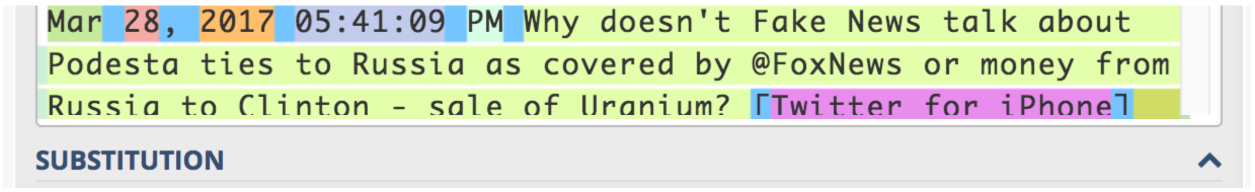
That's right. We immediately create a new group after the "[" and tell it to capture all text until reaching the "]", which we also have to escape.

The last thing we need to do is account for the dummy text at the end of every tweet. The original source had a link back to the original tweet, which is meaningless in this case, but we want to capture it in a group so we can dispose of it. (.+)

So our final expression string looks like this:

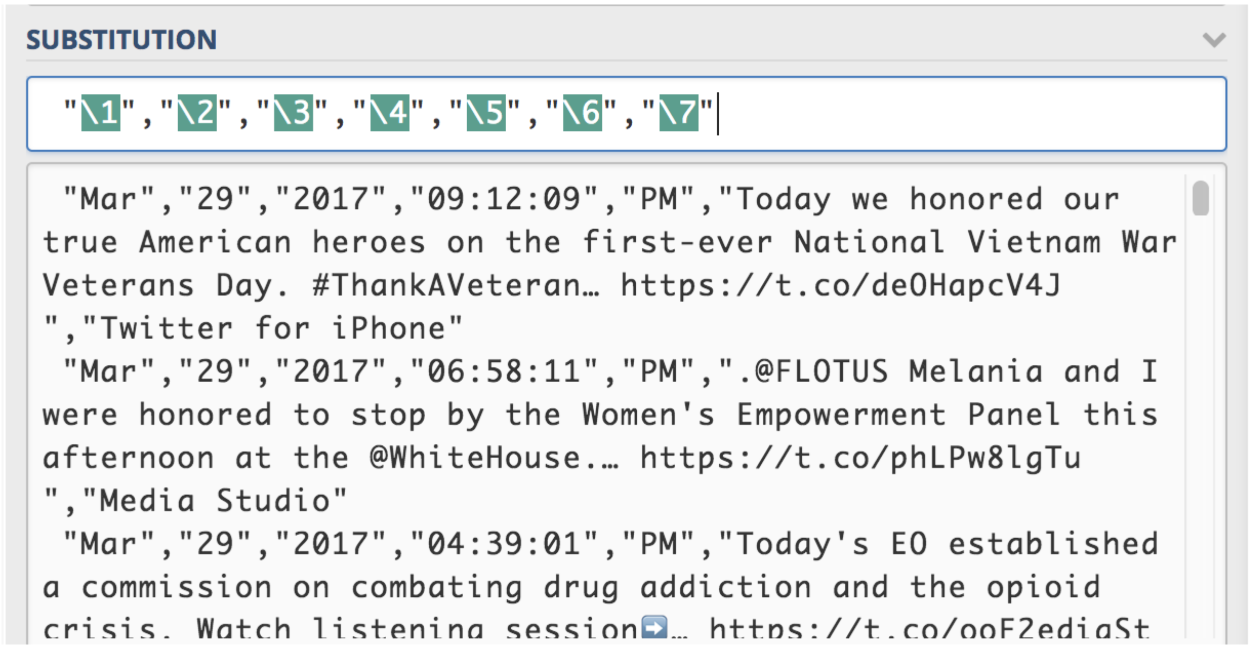
^(\w{3}) (\d+), (\d{4}) (\d\d:\d\d:\d\d) (\w{2}) (.+)\[(.+)\](.+)

Ok, so how do we get our parsed data into a more useful format? On the bottom of the page, click on a bar that says "Substitution".


```
Mar 28, 2017 05:41:09 PM Why doesn't Fake News talk about
Podesta ties to Russia as covered by @FoxNews or money from
Russia to Clinton - sale of Uranium? [Twitter for iPhone]
```

SUBSTITUTION

A box opens up, and in here, we're going to tell Regex how we want our results formatted. Let's create a comma-delimited format with quote marks separating the columns:


SUBSTITUTION

```
"\1","\2","\3","\4","\5","\6","\7"
```

```
 "Mar","29","2017","09:12:09","PM","Today we honored our
true American heroes on the first-ever National Vietnam War
Veterans Day. #ThankAVeteran… https://t.co/deOHapcV4J
","Twitter for iPhone"
 "Mar","29","2017","06:58:11","PM",".@FLOTUS Melania and I
were honored to stop by the Women's Empowerment Panel this
afternoon at the @WhiteHouse.… https://t.co/phLPw8lgTu
","Media Studio"
 "Mar","29","2017","04:39:01","PM","Today's EO established
a commission on combating drug addiction and the opioid
crisis. Watch listening session➡️… https://t.co/ooF2ediaSt
```

Literally speaking, we're telling it to print a ", then group 1, then a quote, comma, quote, then group 2, etc.

You can copy and paste this content into your favorite text editor and save it as a csv file, or paste it directly into a spreadsheet program.

This entire exercise is saved at https://regex101.com/r/XxpV5G/3

For the remainder of the session, clear out the Regex and start again, thinking of other things you might want to capture: Hastags? Links? Etc.

You'll also find more help using Regex online. While we covered only a small portion of what you can do with Regex, I hope this exercise gives you a base from which you can learn more.

Also note that this lesson was inspired by a [tutorial presented at NICAR](#) by Christian McDonald of The Dallas Morning News.