**Learning Journal**

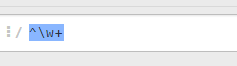
14-09-2020

Christoffer Mondrup Kramer

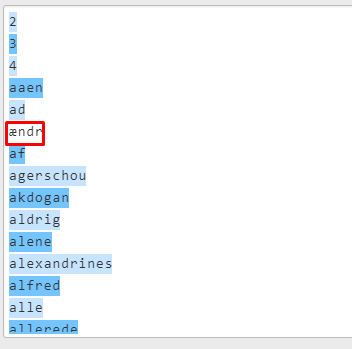
**Assignment 1a:**

Finish the Regex exercises 3 and 4 (see slide 13) as best as you can and document your progress in the Learning Journal. Go back over the library carpentry and do the regex exercises or multichoice quizz to practice further.

I started out by trying to just match the words. I did this with the following expression:



This gave me 538 matches. However, I noticed, that It did not match words containing “æ”:



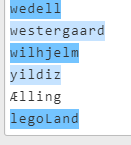
I, therefore, needed to account for Danish letters (æ-ø), which I did with the following expression:



This made it possible to match all words in the list. But, it did not take into case into account. It did not appear to be a problem in this particular word list since all words were written in non-capital letters. However, if the next R word list contained capital letters, I would have to alter the regular expression. I therefore decided to the expression case-insensitive:



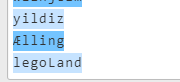
Since all the letters where not capitalized I write two test words (**Æ**lling, lego**L**and), which contained capital letters. But I realized, that I did not get the matches I expected:



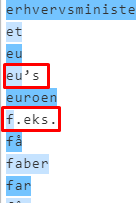
Ælling did not match. I suspected, that this was due to the fact, that it was ÆØÅ. So I read about the \w and realized, that it is case insensitive. The problems was that \w did not consider æøå to be a letter. I therefore did the following:



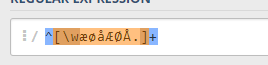
This gave me the expected result:



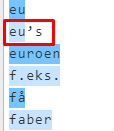
However, while scrolling through the results, I noticed that certain characters were not matched:



I needed to account for special characters. I therefore tried to use the “.”



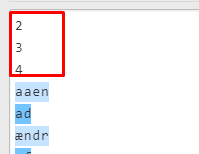
This did not work



The . in brackets apparently only makes an exact match. I therefore moved it out of the brackets:



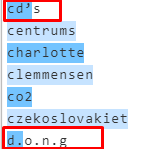
However, now I weren’t matching digits:



I realized that the + should be placed before the . and that it needed to be optional:



This also left out certain results:



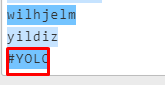
I then researched the + quantifier and realized, that it matches between zero and one time. Not zero and unlimited times. I therefore changed it to:



This caught all words. But something. I thought about adding It in like this aswell, in the case that a word should start with a special character:



This did not change the results, but would be usefull for hashtags, which is widely used on social media. I tested this with the word #YOLO, and got a match:



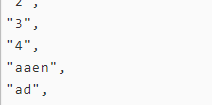
Now I needed to work with the words. I therefore made a group, so I can create variables:



I tried to put “” around the groups and follow it with a comma:



This worked partially, but It did not remove the linebreak:



I first tried to use \b to mark word boundaries, since I suspected that I was matching the line break:



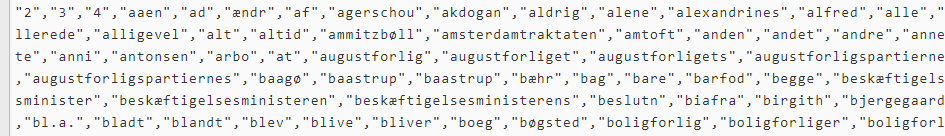
This did not change anything, and I were not matching words that started with ÆØÅ. So I removed I from the front of the Regex, but did not get any new results:



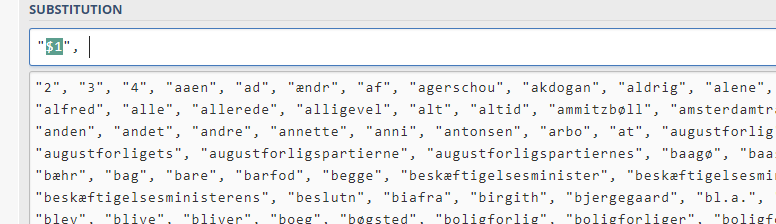
I then tried to group all linebreaks, and not display dem:



This worked:



Now I only needed a space after the comma:



Now it was a Stopword list for R

Regex: (^.\*[\wæøåÆØÅ]+.\*)(\R)

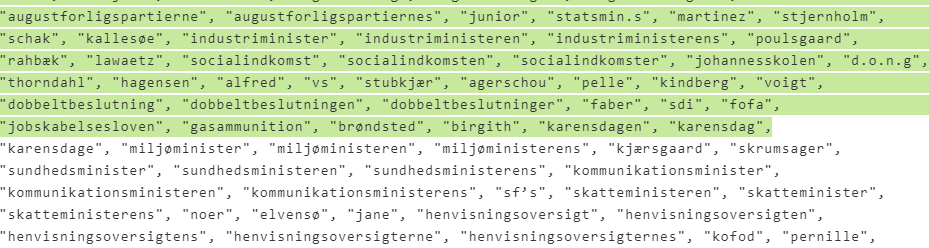
Substitution: "$1",

**Assignment 1 B**

I started with the same regex.



But it did not go as expected.



The first problem was that every word was displayed as one match rather than separate matches. Moreover because of the \R the expression stopped matching after the first linebreak. “

First I tried to use boundary and remove the breakline group, but I got the same results. It treated all words like one match. I suspected, that this was due to the fact that boundaries af specified by spaces, commas and “”.



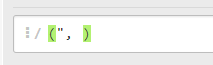
I therefore removed the special characters:



But this created the same problem as previous assignment, where I did not match words containing special characters. Moreover, there where a lot of special characters compared to the previous assignment (such as ë ü). I would therefore risk not catching all words if I tried to specify, exactly which characters I was searching for.

([^", ])

I, therefore, tried something completely different:



Instead of specifying what I want, I tried to capture what I did not want, which is space, “” ,. Then I replaced it with a linebreak:



This almost worked, however, the quotation mark at the start of the word was not captured:



I then added a “” at the end of the word to catch the quationmark after a whitespace:



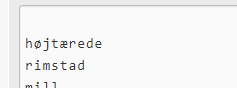
However the words at the beginning and the end of the list contained a “ in front of the word. Moreover a few words did not fit this pattern (probably an entry mistake).

I therefore made a different group to catch “ which did not follow this pattern:



Then I replaced these groups with a line break.

This worked, but left two blank linebreaks (one at the beginning and one at the end):



I tried to fix this but I was unable to do so. But I believed that this was, the best approach, all things considered.

Regex: (“, “)|(“)

Substitution: \R