Homework – week 45

Weekly homework:

Indhold

[1. What regular expressions do you use to extract all the dates in this blurb: http://bit.ly/regexexercise2 and to put them into the following format YYYY-MM-DD? 1](#_Toc87365089)

[2. Write a regular expression to convert the stopwordlist (list of most frequent Danish words) from Voyant in http://bit.ly/regexexercise3 into a neat stopword list for R (which comprises "words" separated by commas, such as http://bit.ly/regexexercise4). Then take the stopwordlist from R http://bit.ly/regexexercise4 and convert it into a Voyant list (words on separate line without interpunction) 2](#_Toc87365090)

[**Exercise 3. Converting a Voyant stopword list to and R stopword list (**A stop-word list for R needs to be formatted as a block of comma-separated words, enclosed in quotations. It should look like the text in exercise 4, ie.: ”a”, ”an”, ”the”, ”of”,...) http://bit.ly/regexexercise3 2](#_Toc87365091)

[**Exercise 4. Converting an R stopword list back to Voyant stopword list** (each word on a new line and without any quotations): http://bit.ly/regexexercise4 4](#_Toc87365092)

[3. In 250 words, answer the following question: "What are the basic principles for using spreadsheets for good data organisation?" 6](#_Toc87365093)

## What regular expressions do you use to extract all the dates in this blurb: <http://bit.ly/regexexercise2> and to put them into the following format YYYY-MM-DD?

Works on a Windows Computer (Lenovo Ideapad): <https://regex101.com/r/Tp68xE/1>

works for everybody: <https://regex101.com/r/ogPKi9/1> And with the correction of the date to YYYY-MM-DD: <https://regex101.com/r/ZwWIMK/1>

**Progress:**

/

(\d{1,2}).(\d{1,2})..?(\d{4})

/

gm

**1st Capturing Group**

(\d{1,2})

\d

 matches a digit (equivalent to [0-9])

{1,2} matches the previous token between 1 and 2 times, as many times as possible, giving back as needed (greedy)

. matches any character (except for line terminators)

**2nd Capturing Group**

(\d{1,2})

\d

 matches a digit (equivalent to [0-9])

{1,2} matches the previous token between 1 and 2 times, as many times as possible, giving back as needed (greedy)

. matches any character (except for line terminators)

.

 matches any character (except for line terminators)

? matches the previous token between zero and one times, as many times as possible, giving back as needed (greedy)

**3rd Capturing Group**

(\d{4})

\d

 matches a digit (equivalent to [0-9])

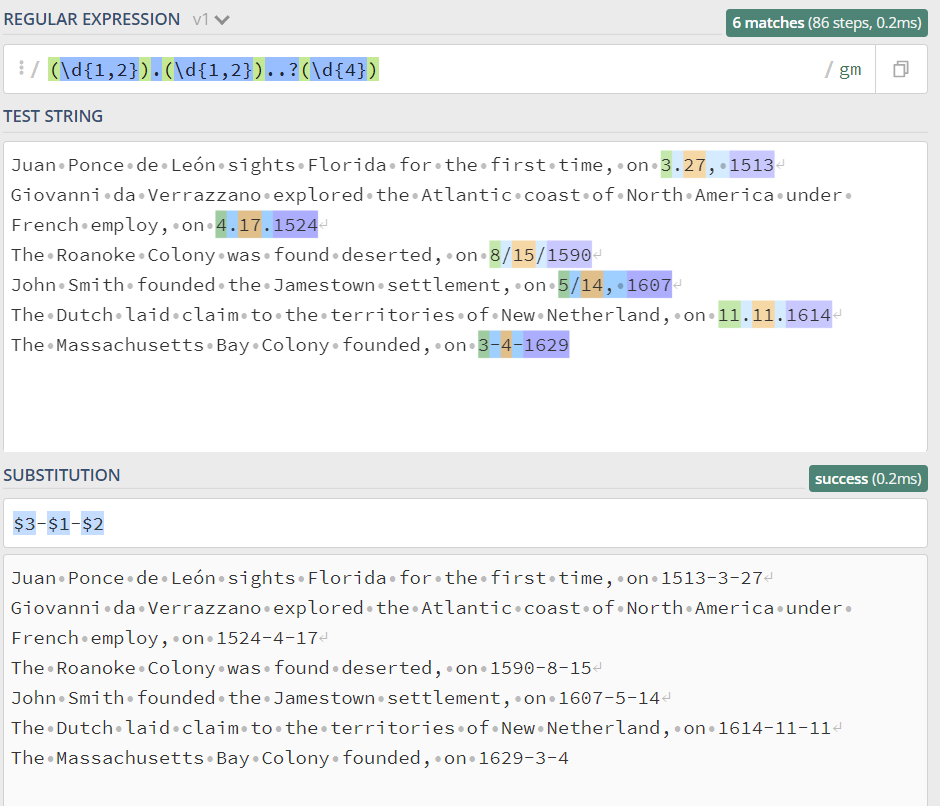
{4} matches the previous token exactly 4 times

**Global pattern flags**

g modifier: **g**lobal. All matches (don't return after first match)

m modifier: **m**ulti line. Causes ^ and $ to match the begin/end of each line (not only begin/end of string)

**Screenshot of the solution with the correction of the date in the bottom of the picture:**



## 2. Write a regular expression to convert the stopwordlist (list of most frequent Danish words) from Voyant in <http://bit.ly/regexexercise3> into a neat stopword list for R (which comprises "words" separated by commas, such as <http://bit.ly/regexexercise4>). Then take the stopwordlist from R <http://bit.ly/regexexercise4> and convert it into a Voyant list (words on separate line without interpunction)

### **Exercise 3. Converting a Voyant stopword list to and R stopword list (**A stop-word list for R needs to be formatted as a block of comma-separated words, enclosed in quotations. It should look like the text in exercise 4, ie.: ”a”, ”an”, ”the”, ”of”,...) <http://bit.ly/regexexercise3>

First try: See link: <https://regex101.com/r/uvfeMg/1>

/

([A-Za-zæøåüé234’]+)|([\n])

/

gm

**1st Alternative**

([A-Za-zæøåüé234’]+)

**1st Capturing Group**

([A-Za-zæøåüé234’]+)

**Match a single character present in the list below**

[A-Za-zæøåüé234’]

+ matches the previous token between one and unlimited times, as many times as possible, giving back as needed (greedy)

A-Z matches a single character in the range between A (index 65) and Z (index 90) (case sensitive)

a-z matches a single character in the range between a (index 97) and z (index 122) (case sensitive)

æøåüé234’

 matches a single character in the list æøåüé234’ (case sensitive)

**2nd Alternative**

([\n])

**2nd Capturing Group**

([\n])

**Match a single character present in the list below**

[\n]

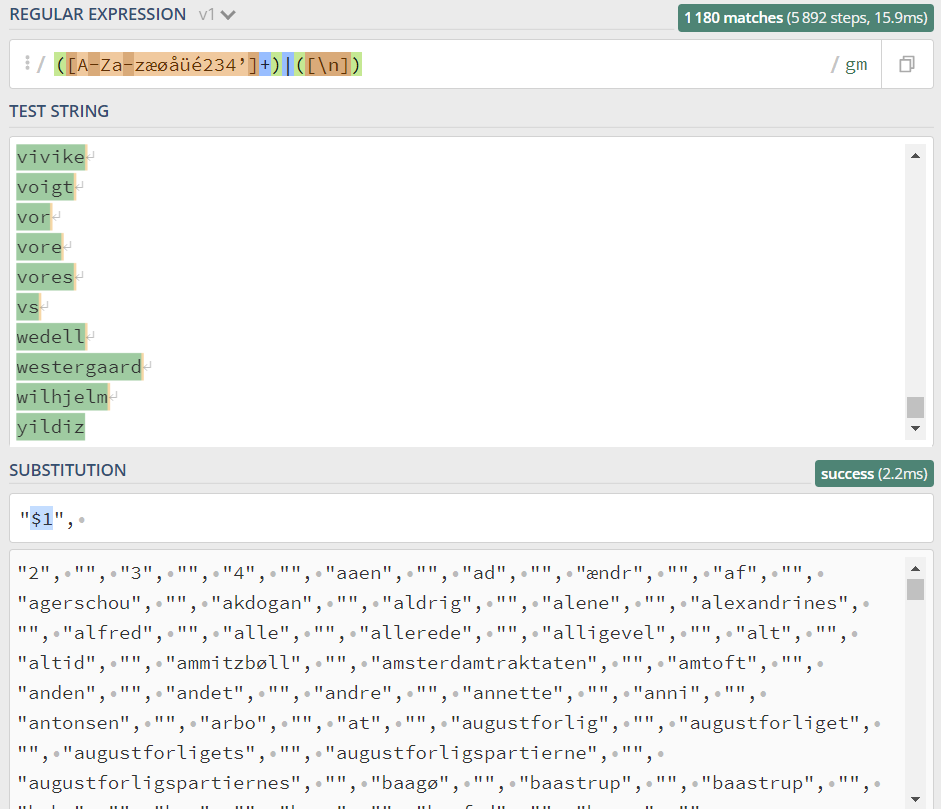
\n matches a line-feed (newline) character (ASCII 10)

**Global pattern flags**

g modifier: **g**lobal. All matches (don't return after first match)

m modifier: **m**ulti line. Causes ^ and $ to match the begin/end of each line (not only begin/end of string)

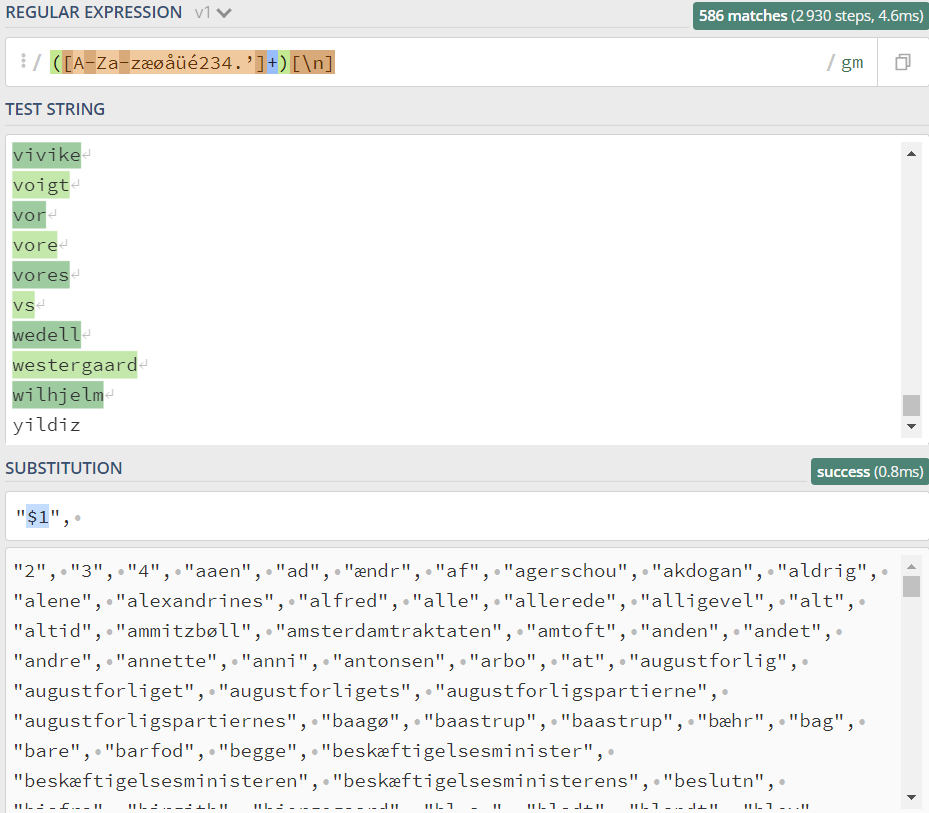
**Screenshot of the result. There is a flaw, because the empty spaces are also marked with “”,. Which it shouldn’t be.**



#### Second try:

We removed the | from our code and found that it did not take the empty spaces into the “”, as we got before.

The link to the successful try: <https://regex101.com/r/68iU4D/1>



### **Exercise 4. Converting an R stopword list back to Voyant stopword list** (each word on a new line and without any quotations): <http://bit.ly/regexexercise4>

**Process:**

(["])|([", ])|([A-Za-zøæüåé.]+)

/

gm

**1st Alternative**

(["])

**1st Capturing Group**

(["])

**Match a single character present in the list below**

["]

" matches the character " with index 3410 (2216 or 428) literally (case sensitive)

**2nd Alternative**

([", ])

**2nd Capturing Group**

([", ])

**Match a single character present in the list below**

[", ]

",

 matches a single character in the list ",  (case sensitive)

**3rd Alternative**

([A-Za-zøæüåé.]+)

**3rd Capturing Group**

([A-Za-zøæüåé.]+)

**Match a single character present in the list below**

[A-Za-zøæüåé.]

+ matches the previous token between one and unlimited times, as many times as possible, giving back as needed (greedy)

A-Z matches a single character in the range between A (index 65) and Z (index 90) (case sensitive)

a-z matches a single character in the range between a (index 97) and z (index 122) (case sensitive)

øæüåé.

 matches a single character in the list øæüåé. (case sensitive)

**Global pattern flags**

g modifier: **g**lobal. All matches (don't return after first match)

m modifier: **m**ulti line. Causes ^ and $ to match the begin/end of each line (not only begin/end of string)

**Screenshot of the result:**



See link of the saved code: <https://regex101.com/r/3BJS5W/1>

## 3. In 250 words, answer the following question: "What are the basic principles for using spreadsheets for good data organisation?"

Spreadsheets are a software tool which is used for storing data, entering data, viewing data, and analyzing data. Spreadsheets are good tool for organizing data, however, if it is not done correctly it can confuse more than help. In the following I will present some of the basic principles for using spreadsheets for good data organization. The first good rule for sufficient data organization is to be consistent in choosing titles, words, phrases and date formats. Instead of using different words or shortenings for words, then be consistent and choose only one word to use. Consistency in writing dates is also important. The most recommended is using the “ISO 8601”-standard, yyyy-mm-dd. That will make it easier for the next person to figure out the data in the spreadsheet and which spreadsheet is used for what. Paying extra attention to naming things such as filenames and variable names is also an important principle to good data organization. One must avoid spaces in titles and names as well as special characters. And instead of making long filenames, make sure to make them short but meaningful.

A good principle is also to make sure to leave no empty cells. Instead of leaving cells empty write “NA” or “et al.” to prevent confusion over whether it’s a flaw or it was left empty on purpose. As well as avoiding empty cells, on should also avoid putting more than one thing in a cell.

The last good principle is to make sure to have backups and saving the Data in Plan Text Files that doesn’t require a special software. CVS is recommended for saving, especially if the file is simple and containing primary data. If one sticks to the basic principles as mentioned, organizing spreadsheets will be much more efficient.