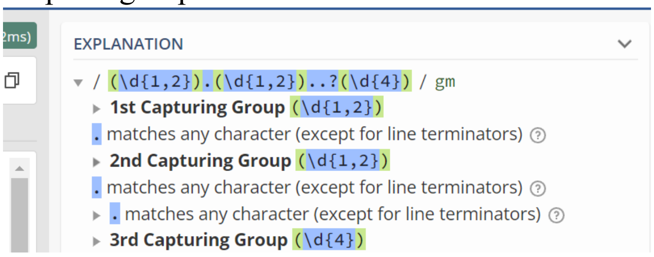
Homework 08-11-2022:

Task 1: extracting the dates

1. I write: **\d{1,2}.\d{1,2}..?\d{4}** in the regular expression bar
    a. \d: to only get digits
    b. I use {} to indicate a value
    c. "." Because it can mean any character
    d. "?" because it matches the preceding character (*in this case, "."*) 0-1 times

2. To change the format from: **DD-MM-YYYY** to: **YYYY-MM-DD** I do the following:
    a. I surround the Regular expression in "( )": **(\d{1,2}).(\d{1,2})..?(\d{4})**
    - By doing this, I can *divided/sorted* the dates from the months and years

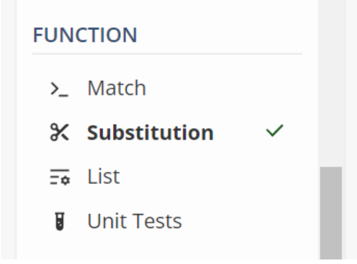
    b. After I have placed a set of "()" each one of them, I have now made them into three "caption groups".



EXPLANATION

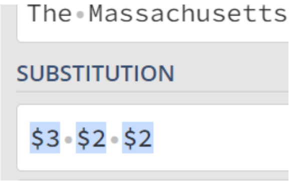
```
/ (\d{1,2}).(\d{1,2})..?(\d{4}) / gm
  1st Capturing Group (\d{1,2})
  . matches any character (except for line terminators)
  2nd Capturing Group (\d{1,2})
  . matches any character (except for line terminators)
  . matches any character (except for line terminators)
  3rd Capturing Group (\d{4})
```

…then I go over to the left side of the textbox, then select the "substitution" function.



FUNCTION

- Match
- **Substitution** ✓
- List
- Unit Tests



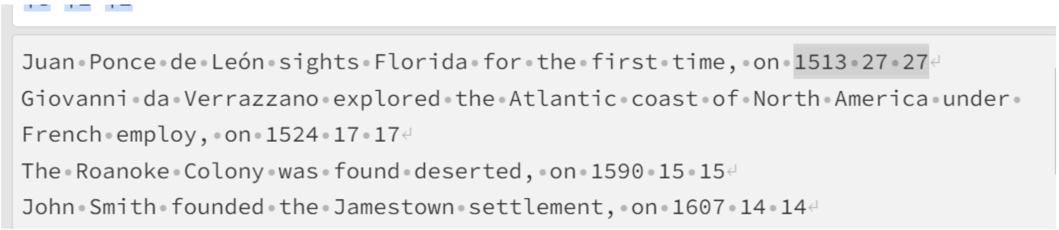
The Massachusetts

SUBSTITUTION

```
$3 $2 $2
```

    c. In the "substitution" function I write the following: $3 $2 $2
    - Here the "$3" means the third "caption group", which translates to the: (\d{4}) ← the last one in the regular expression, which is the year
    - The following two "$2" referes to the month and dates (second and first capturing group)

    d. Then, after the three $ is written, the dates in the text is transformed to this:

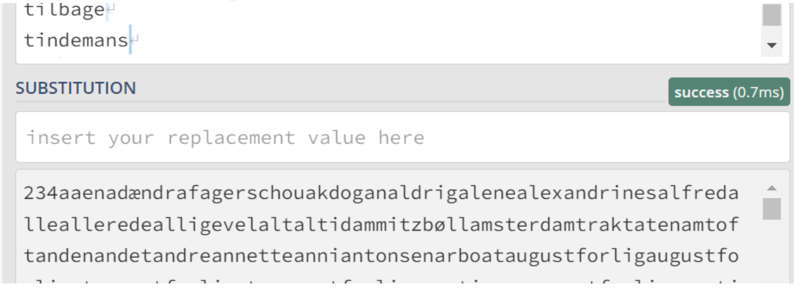


```
Juan Ponce de León sights Florida for the first time, on 1513 27 27
Giovanni da Verrazzano explored the Atlantic coast of North America under
French employ, on 1524 17 17
The Roanoke Colony was found deserted, on 1590 15 15
John Smith founded the Jamestown settlement, on 1607 14 14
```

    - I have highlighted an example of the change

Task 2: converting stopwordlists via a regular expression
converting from Voyant to R (*from list to text)*

- After I insert the Voyant StopWordList to Regex101, I write in the Regular expression bar: \n
- The reason for this is, that "\n" means "newline", and therefore it marks the ending in every sentence. This doesn't do much before you open the "substitution" function.
- In the substitution function, you can now see, that the Voyant list is converted to one whole chunk of text, which doesn't make that much sense, due to the lack of commas or period



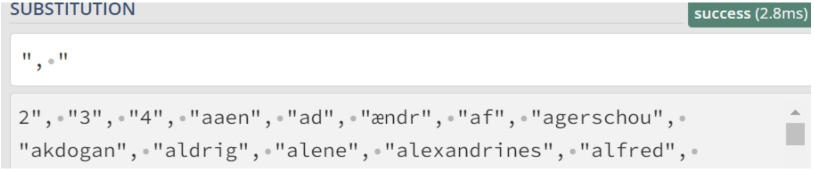
```
tilbage
tindemans
```

SUBSTITUTION success (0.7ms)

```
insert your replacement value here
```

```
234aaenadændrafagerschouakdoganaldrigalenealexandrinesalfreda
llealleredealligevelaltaltidammitzbøllamsterdamtraktatenamtof
tandenandetandreannetteanniantonsenarboataugustforligaugustfo
```

- to separate the text into separate words, I then write the following in the bar: **"**. **"**



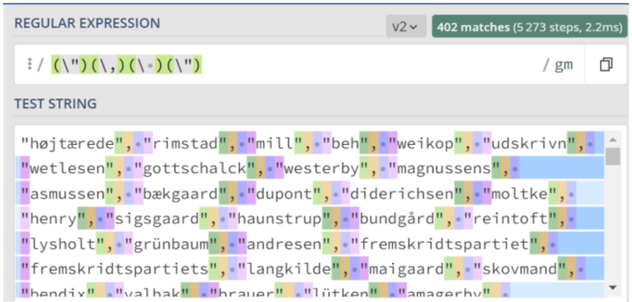
SUBSTITUTION success (2.8ms)

```
", "
```

```
2", "3", "4", "aaen", "ad", "ændr", "af", "agerschou", 
"akdogan", "aldrig", "alene", "alexandrines", "alfred", 
```

a. converting from R to Voyant

- To then convert from R (text) to Voyant (list), there is two different ways to write the Regular Expression: the first one is where I highlight each character in the textbox: **(\")(\,)(\ )(\")**



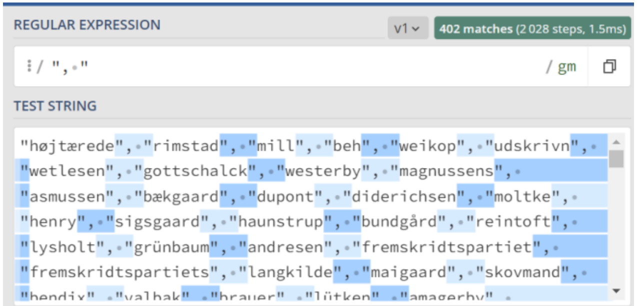
REGULAR EXPRESSION v2 402 matches (5 273 steps, 2.2ms)

```
/ (\")(\,)(\ )(\") / gm
```

TEST STRING

```
"højtærede", "rimstad", "mill", "beh", "weikop", "udskrivn", 
"wetlesen", "gottschalck", "westerby", "magnussens", 
"asmussen", "bækgaard", "dupont", "diderichsen", "moltke", 
"henry", "sigsgaard", "haunstrup", "bundgård", "reintoft", 
"lysholt", "grünbaum", "andresen", "fremskridtspartiet", 
"fremskridtspartiets", "langkilde", "maigaard", "skovmand", 
"hendix", "valhak", "brauer", "lütken", "amagerby", 
```

This way it however the more complex way of doing it, and there is a faster and more simple way, that is: **", "**



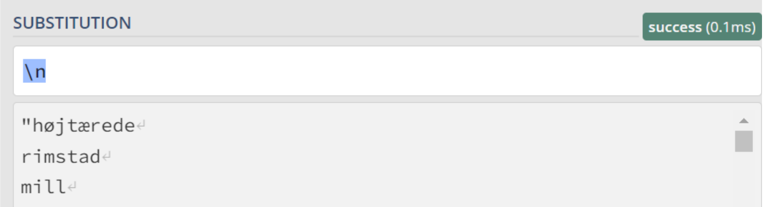
REGULAR EXPRESSION v1 402 matches (2 028 steps, 1.5ms)

```
/ ", " / gm
```

TEST STRING

```
"højtærede", "rimstad", "mill", "beh", "weikop", "udskrivn", 
"wetlesen", "gottschalck", "westerby", "magnussens", 
"asmussen", "bækgaard", "dupont", "diderichsen", "moltke", 
"henry", "sigsgaard", "haunstrup", "bundgård", "reintoft", 
"lysholt", "grünbaum", "andresen", "fremskridtspartiet", 
"fremskridtspartiets", "langkilde", "maigaard", "skovmand", 
"hendix", "valhak", "brauer", "lütken", "amagerby", 
```

- But when it comes to converting to a list in the “substitution bar”, it’s the same procedure: **\n**

  ```
  \n
  ```

  ```
  "højtærede
  rimstad
  mill
  ```

  - the reason for writing \n (which by name is called “newline”), is because this general token’s function is creating a new line. It separates the words by lines, and by this creating a Wordlist, that is functional in Voyant.
- As you can see under the Substitution bar, it’s not possible to remove the first apostrophe from *Højtærede*, and this is something that has to be removed manually
- You can say that the conversion procedure from Voyant to R and vice versa is the same but written different. From Voyant to R you write \n in the regular expression bar, where you the other way around write \n in the substitution bar

Task 3: "What are the basic principles for using spreadsheets for good data organization?"

There are twelve basic principles for using spreadsheets to achieve good data organization, some more important than others. The first, and most important one, is being *consistent*. And this applies to every single step and procedure in data management and organization. By making sure the data management is consistence from the start, there is a lot of time saved in the end, even though it at the moment feels like a tedious and time consuming.[1]

The consistency applies to e.g., names, files, formats, dates[2] in addition to *consistency* is *missing data and empty cells*. When there is no data available, it’s better to write “N/A”, than to leave cells empty. By leaving them empty, not everyone will know why. In addition to no empty cells is *only one thing in each cell*. Another important principle when it comes to organization, is keeping a data dictionary in a separate file. By doing this, you will never not know what the different variables mean, and what they are used for. Also, it makes it a lot easier for others to understand and use the project.[3]

Lastly, not changing and calculating in the raw data is crucial, when it comes to keeping the data organized, and accessible. By editing the raw data, the risk of errors and “junk data” is bigger, and this can be damaging for the project. Therefore, it’s best to write-protect it, back it up and don’t touch it.[4]

[1] Karl W. Broman & Kara H. Woo (2018) Data Organization in Spreadsheets, The American Statistician, 72:1, 2-10, DOI: 10.1080/00031305.2017.1375989 page 2

[2] Page 3. Names and dates are two separate principles,

[3] Page 6.

[4] Page 7