Scraping_pdf

First, install and call some general packages you will likely need.

library(dplyr)

library(tidytext)

Scraping pdf files

In another tutorial we scraped some news data from the internet, we also pulled data from reddit.

However, often we are interested in systematizing and analysing a set of data that exist in document form such as party manifestos or academic articles, or human rights reports, or judges opinions. Often these data are available in pdf format. This short tutorial explains how to extract data from pdf files.

For more see: Reading PDF Files

0: Set your working directory

For r to know where to find your files it is useful to set your working directory at the beginning of the tutorial to the folder you will be working in where your pdf files are.

setwd("/path/nameofDirectory")

1: Package for reading pdf

For information about this package see: pdftools

```
#install.packages("pdftools") #Text Extraction, Rendering and Converting of PDF Documents.
library(pdftools)
```

```
Using poppler version 23.04.0
```

2: Creating a list of your pdf files

Create a vector of pdf file names from the pdf files in your directory (remember to set the working directory to the directory where you are keeping your files)

```
pdf_files <- list.files(pattern = "pdf$")
#The list.files is a very helpful basic R function.  For more see https://www.r-bloggers.c

pdf_files
```

```
[1] "13-1314_ook3.pdf"          "14-46_1b5p.pdf"
[3] "14-7955_1823.pdf"          "93-95_8-21-YUGOSLAVIA.pdf"
[5] "ScrapingPdf.pdf"
```

(See R base functions, for a full list.)

3: Extract the text from the pdf

Next we extract all the text from all of the pdf_files that we listed in our prior object.

```
pdf_content <- lapply(pdf_files, pdf_text) #The lapply() function from base R takes the li
#The pdf_text function from the pdftools package extracts the text from all the pdfs in ou
```

We can verify how many documents in pdf_content were read:

```
length(pdf_content)
```

```
[1] 5
```

Count the number of pages in each document in pdf_content

```
[[1]]
[1] 68

[[2]]
[1] 106

[[3]]
[1] 70

[[4]]
[1] 237

[[5]]
[1] 7
```

To see the pdf_content

```
#pdf_content  #lets not do this because it takes forever and creates a very large file but
```

4: Convert to document-term-matrix

Next we want to convert the list text to a matrix

There are a few options here. Conventional matrixes are either in the form of document term matrix - where documents define the rows and words or terms define the columns and the occurance of each term is either counted as a 0 or a 1 in each document - or or term document where terms define the rows and documents define the columns.

```
#additional packages
library(tidyverse)
```

```
-- Attaching core tidyverse packages ------------------------ tidyverse 2.0.0 --
v forcats   1.0.0     v readr     2.1.4
v ggplot2   3.4.3     v stringr   1.5.0
v lubridate 1.9.2     v tibble    3.2.1
v purrr     1.0.2     v tidyr     1.3.0
-- Conflicts ------------------------------------------ tidyverse_conflicts() --
x dplyr::filter() masks stats::filter()
x dplyr::lag()    masks stats::lag()
i Use the conflicted package (<http://conflicted.r-lib.org/>) to force all conflicts to becor
```

```
library(tm) #A framework for text mining applications within R.  See https://cran.r-projec
```

```
Loading required package: NLP

Attaching package: 'NLP'

The following object is masked from 'package:ggplot2':

    annotate
```

```
pdf_converted <- Corpus(VectorSource(pdf_content)) %>%
          DocumentTermMatrix()

pdf_converted
```

```
<<DocumentTermMatrix (documents: 5, terms: 31683)>>
Non-/sparse entries: 37987/120428
Sparsity           : 76%
Maximal term length: 113
Weighting          : term frequency (tf)
```

5: Convert to a tidy format

DocumentTermMatrixes are very large and computationally intensive to work with. Tibbles organize the text in a vector defined by a vector of documents and vector of words in each documen so there are no 0s. We will use tibbles here because of computational efficiency.

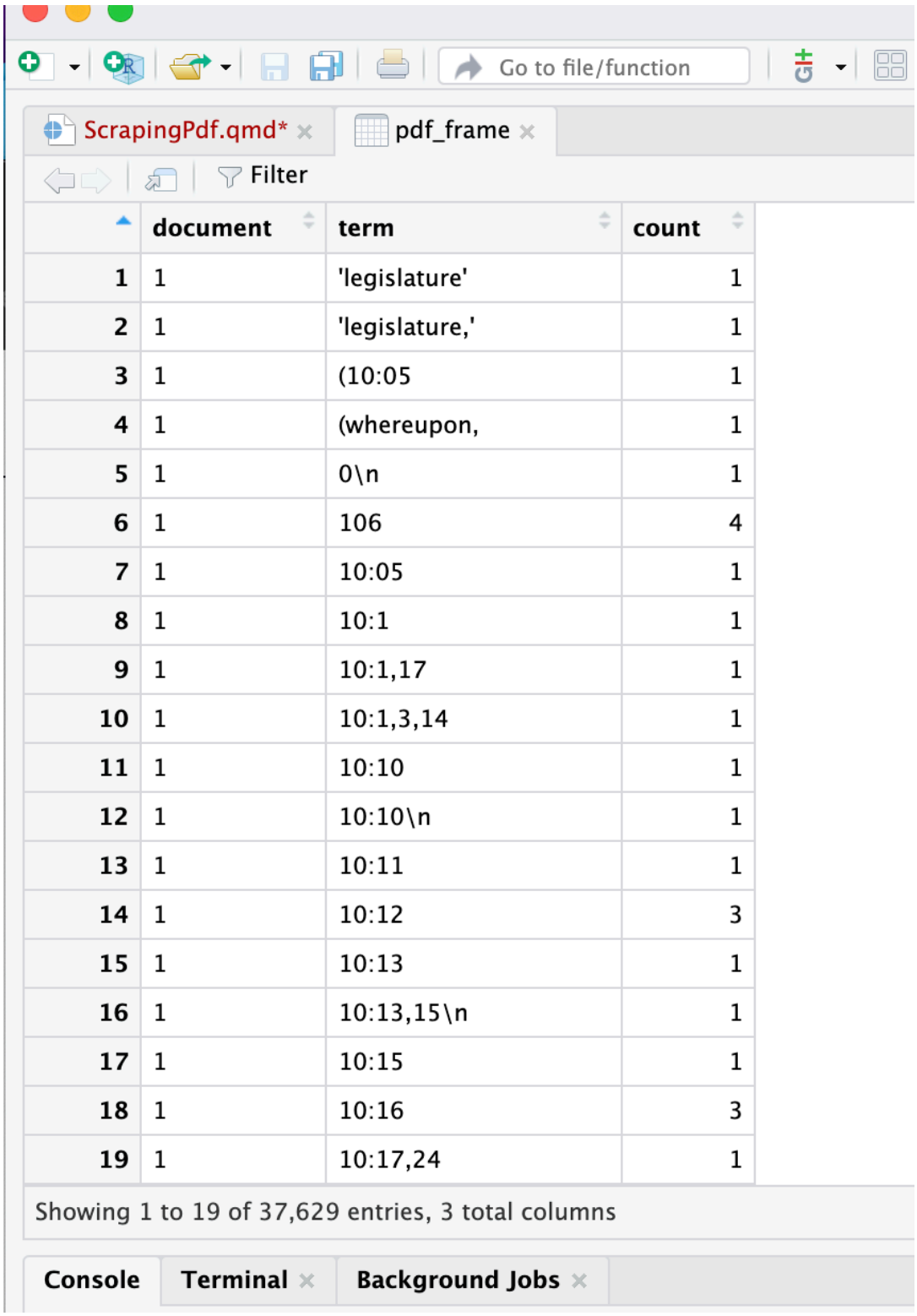
Therefore a final step is to tidy our document matrix into a tibble frame for analysis

```
# see https://cran.r-project.org/web/packages/tidytext/vignettes/tidying_casting.html
library(dplyr)
library(tidytext)

pdf_frame = tidy(pdf_converted)

pdf_frame
```

```
# A tibble: 37,987 x 3
   document term             count
   <chr>    <chr>            <dbl>
```

```
 1 1        "'legislature'"      1
 2 1        "'legislature,'"     1
 3 1        "(10:05"             1
 4 1        "(whereupon,"        1
 5 1        "0\\n"               1
 6 1        "106"                4
 7 1        "10:05"              1
 8 1        "10:1"               1
 9 1        "10:1,17"            1
10 1        "10:1,3,14"          1
# i 37,977 more rows
```

What you should see now is that the data has been saved as a dataframe with one variable accounting for the document number (id) and the next variable listing words in that document and the third variable counting how often that word occurs.



	document	term	count
1	1	'legislature'	1
2	1	'legislature,'	1
3	1	(10:05	1
4	1	(whereupon,	1
5	1	0\n	1
6	1	106	4
7	1	10:05	1
8	1	10:1	1
9	1	10:1,17	1
10	1	10:1,3,14	1
11	1	10:10	1
12	1	10:10\n	1
13	1	10:11	1
14	1	10:12	3
15	1	10:13	1
16	1	10:13,15\n	1
17	1	10:15	1
18	1	10:16	3
19	1	10:17,24	1

Clearly

there remains some cleaning of this text to be done :)

Cleaning

```
pdf_frame1 =pdf_frame %>%
  filter(!grepl("[0-9]+", term)) #NOTE: filters out non-grepl (string) digits from 0-9 in

save(pdf_frame1, file = "pdf_frame1.RData")

#do some more cleaning
pdf_frame1 <-pdf_frame1 %>%
  filter(!grepl("\"", term))

pdf_frame1 <-pdf_frame1 %>%
  filter(!grepl("'", term))


save(pdf_frame1, file = "pdf_frame1.RData")
```

and so on.