Text Analysis 1 - Demo Notebook

Dani Madrid-Morales

Demo 0 - Installing packages, and setting up the environment

The goal of this demo is to offer a gentle introduction to text analysis using `quanteda`, an R package that allows you to transform a collection of texts into a corpus, "clean" the documents, re-shape them, describe them, and create simple visualizations. The first step to any project in R is to install and load any required packages. This process is best managed with the package `pacman`.

If this is the first time you are using R, or if you have never used `pacman`, you will first need to install it. Notice that, in the code below, the line `install.packages("pacman")` has a hashtag (#) at the beginning; this indicates that this line of code should NOT be executed/ran. So, if you have never used `pacman` before, delete the hashtag in front of this line of code before running the chunk below.

Remember that you only need to install packages once, but you need to load them every time you start R.

For this demo, we will use a dataset of academic journal article abstracts ranging from 2000-2023 and published in the *Housing Studies* journal. For more information about the dataset, check this link.

`quanteda`, like many other R packages, is constantly being developed and improved, so it changes regularly. New features are being added constantly, and that means that, sometimes, functions or function parameters are deprecated or renamed. This means that you may encounter many code examples, StackOverflow questions, and websites/tutorials with outdated documentation, etc. that include functions or options that have been deprecated or renamed. Keep that in mind when you write your code and look for help online, as some functions might have become obsolete.

Fortunately, `quanteda` has some excellent documentation, with lots of examples.

Demo 1 - Gentle Introduction to `quanteda`

You can think of `quanteda` a Swiss army knife for computational text analysis. It is a package that is able to do the most common operations needed for a project that uses computational methods to analyse texts.

The first step of using `quanteda` is to import the data. Using the `readtext` package is probably the most convenient way to import texts into R (and `quanteda`). `readtext` was developed by the same team that developed `quanteda`. We can import data from a CSV file, a collection of Word documents, a series of text files… There's, in fact, tons of different ways to store text data, and many can be imported using `readtext`. You can read how to handle some common situations in the documentation of the package. To load the documentation, you can use the command `?readtext`.

The dataset we will be using in this workshop is in a CSV format with 27 columns. One of the columns includes the text of each abstract (one abstract per row), and the additional columns include "metadata" (i.e. additional information about each document, such as the author, the date…). In `quanteda`, we call these "metadata" docvars, from the term document variables.

IMPORTANT : If you have downloaded the dataset to your computer, make sure it is saved in a folder called "data", which should be in the SAME folder as this document. The code below will not work if the organisation of files in your folders is not correct.

To import data from a csv file, `readtext` has a single function, `readtext` that requires us to specify the name of the column in the csv file that has the text data. In our case, the text we are interested in is in a column called "Abstract".

```
df_abstracts <- readtext(file = "data/housing_scopus.csv",
                         text_field = "Abstract")
```

The resulting object is a `readtext` object with 28 variables, including:

- `text` which includes the full text of the abstract for each paper
- `docid` which will be used by `quanteda` to identify each text in our dataset with a unique KID

The rest of variables are the *docvars* (or metadata) that we were referring to earlier. *docvars* are very useful, as they allows us to separate the corpus (or group it) according to some theoretically-driven characteristics of our data. This will allow us, for example, to compare different groups in our corpora.

Before we proceed any further, let's have a look at our data. If you inspect it closely, you'll see that there's some missing values in our data in variables such as `united_kingdom` and `united_states`. These are variables that tell us whether the author of a paper is UK based or US based. Missing values can complicate our analysis and, in some cases, it might make sense to remove them altogether. This is what we are going to do right now.

```
# Remove rows that have missing values
df_abstracts %<>%
  drop_na(united_states) %>%
  drop_na(united_kingdom)
```

This have left us with 1422 abstracts instead of the 1500 we started with.

Right now, our dataset is in a table-like format (we call this a dataframe in R). The next thing we need to do is to transform this dataframe into a corpus object in `quanteda`. To do so, all we need is the `corpus` function.

```
# Create a corpus with quanteda
hs_corpus <- corpus(df_abstracts)

# We can see what's inside our corpus using the `summary` command
# The default number of entries to display is 100.
# We can lower the number by adding the number we want to display
summary(hs_corpus, 3)
```

```
## Corpus consisting of 1422 documents, showing 3 documents:
##
##                  Text Types Tokens Sentences
##  housing_scopus.csv.1   114    191         7
##  housing_scopus.csv.2   124    200         9
##  housing_scopus.csv.3   112    172         8
##                                                            Authors
##  Chisholm E., Olin C., Randal E., Witten K., Howden-Chapman P.
##                           Brookfield K., Dimond C., Williams S.G.
##                                   Elkins M., Farrell L., Fry J.M.
##                                                       Author.s..ID
##  57188823391;57364756200;35778943100;6601989568;7003417304;
##                         55794991400;58055395900;58055933200;
##                           55964784900;7006981468;7201780339;
##                                                                                                                Title
##                        Placemaking and public housing: the state of knowledge and research priorities
##                        Selling city centre flats in uncertain times: findings from two English cities
##   Homelessness and housing insecurity among youth in Australia: sequence analysis of housing careers
##   Year    Source.title Volume Issue Art..No. Page.start Page.end Page.count
##   2023 Housing Studies     NA             NA                              NA
##   2023 Housing Studies     NA             NA                              NA
##   2023 Housing Studies     NA             NA                              NA
##   Cited.by                           DOI
##         NA 10.1080/02673037.2023.2206799
##         NA 10.1080/02673037.2023.2203095
##         NA 10.1080/02673037.2023.2203081
##
##   https://www.scopus.com/inward/record.uri?eid=2-s2.0-85158979413&doi=10.1080%2f02673037.2023.2206799
##   https://www.scopus.com/inward/record.uri?eid=2-s2.0-85158099509&doi=10.1080%2f02673037.2023.2203095
##   https://www.scopus.com/inward/record.uri?eid=2-s2.0-85153594708&doi=10.1080%2f02673037.2023.2203081
##
##   Department of Public Health, University of Otago, Wellington, New Zealand; SHORE and Whariki Resear
##     Department of Environment and Geography, University of York, York, United Kingdom; Department of 
##                                                         School of Economics, Finance & Marketing, R
##
##   Chisholm, E., Department of Public Health, University of Otago, Wellington, New Zealand; Olin, C., 
##
##
##                                                                                                            Author.Ke
##           housing regeneration; place attachment; Placemaking; public housing; public space; sense of
##   city centre housing; comparative research; documentary research; Housing markets; real estate adver
##                                                     homelessness; Housing; Journeys Home; sequence analysis;
##   Index.Keywords Document.Type Publication.Stage Open.Access Source
##                        Article  Article in Press          NA Scopus
##                        Article  Article in Press          NA Scopus
##                        Article  Article in Press          NA Scopus
##                  EID quant united_kingdom united_states
##   2-s2.0-85158979413     1              0             0
##   2-s2.0-85158099509     0              1             0
##   2-s2.0-85153594708     1              0             0
```

Notice how quanteda differentiates between “types” (unique words) and “tokens” (all words) in a document. To be more precise, both type and token also include punctuation and special symbols. When we create a corpus, we are also given a summary of the number of sentences each document has. This would allow us, if we wanted to, to change the unit of analysis from document level to sentence level.

Depending on what it is that we are studying, we might determine that, the best documentary unit (i.e. how do we want to break down the corpus) is a sentence, or a paragraph, or the full document. We can make this transformations easily with the `corpus_reshape` command.

```r
# You could change the unit of analysis (defaults to "document") to sentences
hs_sent_corpus <- corpus_reshape(hs_corpus, to = 'sentences')
ndoc(hs_sent_corpus)
```

```
## [1] 10088
```

```r
ndoc(hs_corpus)
```

```
## [1] 1422
```

```r
summary(hs_sent_corpus, 3)
```

```
## Corpus consisting of 10088 documents, showing 3 documents:
##
##                    Text Types Tokens Sentences
##  housing_scopus.csv.1.1    20     20         1
##  housing_scopus.csv.1.2    27     34         1
##  housing_scopus.csv.1.3    34     45         1
##                                                        Authors
##  Chisholm E., Olin C., Randal E., Witten K., Howden-Chapman P.
##  Chisholm E., Olin C., Randal E., Witten K., Howden-Chapman P.
##  Chisholm E., Olin C., Randal E., Witten K., Howden-Chapman P.
##                                                Author.s..ID
##  57188823391;57364756200;35778943100;6601989568;7003417304;
##  57188823391;57364756200;35778943100;6601989568;7003417304;
##  57188823391;57364756200;35778943100;6601989568;7003417304;
##                                                                          Title
##  Placemaking and public housing: the state of knowledge and research priorities
##  Placemaking and public housing: the state of knowledge and research priorities
##  Placemaking and public housing: the state of knowledge and research priorities
##  Year    Source.title Volume Issue Art..No. Page.start Page.end Page.count
##  2023 Housing Studies     NA             NA                             NA
##  2023 Housing Studies     NA             NA                             NA
##  2023 Housing Studies     NA             NA                             NA
##  Cited.by                          DOI
##        NA 10.1080/02673037.2023.2206799
##        NA 10.1080/02673037.2023.2206799
##        NA 10.1080/02673037.2023.2206799
##
##  https://www.scopus.com/inward/record.uri?eid=2-s2.0-85158979413&doi=10.1080%2f02673037.2023.2206799
##  https://www.scopus.com/inward/record.uri?eid=2-s2.0-85158979413&doi=10.1080%2f02673037.2023.2206799
##  https://www.scopus.com/inward/record.uri?eid=2-s2.0-85158979413&doi=10.1080%2f02673037.2023.2206799
##
##  Department of Public Health, University of Otago, Wellington, New Zealand; SHORE and Whariki Resear
##  Department of Public Health, University of Otago, Wellington, New Zealand; SHORE and Whariki Resear
##  Department of Public Health, University of Otago, Wellington, New Zealand; SHORE and Whariki Resear
##
##  Chisholm, E., Department of Public Health, University of Otago, Wellington, New Zealand; Olin, C., 
##  Chisholm, E., Department of Public Health, University of Otago, Wellington, New Zealand; Olin, C., 
##  Chisholm, E., Department of Public Health, University of Otago, Wellington, New Zealand; Olin, C., 
##                                                                                Author.Keywords
##  housing regeneration; place attachment; Placemaking; public housing; public space; sense of place
##  housing regeneration; place attachment; Placemaking; public housing; public space; sense of place
##  housing regeneration; place attachment; Placemaking; public housing; public space; sense of place
##  Index.Keywords Document.Type Publication.Stage Open.Access Source
##                       Article  Article in Press          NA Scopus
##                       Article  Article in Press          NA Scopus
##                       Article  Article in Press          NA Scopus
##                    EID quant united_kingdom united_states
##  2-s2.0-85158979413     1              0             0
##  2-s2.0-85158979413     1              0             0
##  2-s2.0-85158979413     1              0             0
```

Determining the best documentary unit really depends on your RQs or Hs. In our case, we are not interested in the granularity provided by a sentence-by-sentence analysis, so we will keep the abstract as the unit of analysis.

In summary, in this first part we have done 4 things:

1. We have loaded all the required packages for our project.
2. We have imported texts from a CSV file using `readtext`.
3. We have created a corpus of 1500 documents with metadata (*docvars*) using `quanteda`.
4. We have tested how to change the unit of analysis with the command `corpus_reshape`.

Demo 2 - Text pre-processing using `quanteda`

It is during the pre-processing stage in a computational text analysis project that we make some of the most consequential decisions. This is the stage in which we decide what features (i.e., words) to include, and what features to transform. This is also the stage that tends to bring most human involvement as many of these decisions will need to be made (and justified) by the researcher(s). Our choices during this stage will have an impact on the outcome of the analysis.

While there isn't a single best workflow to pre-process our data, we generally follow the same four steps:

1. Tokenize - we break down each text in the corpus into tokens.
2. Remove punctuation, capitalization and/or any other features we are not interested in.
3. Discard stopwords - we can use existing lists, or create our own lists.
4. Stem and/or lemmatize our corpus, depending on our needs.

In some cases, you might need/want to skip some of them (e.g. sometimes, capitalized words matter, and therefore we would not lowercase our corpus).

The function `tokens` is used to transform a corpus (a collection of documents are their metadata), to a tokenized version of the corpus (that is, a corpus that has been broken down into features). The command `tokens` has a bunch of options that you can explore using the command `?tokens`. These options allow you to decide what features you include/keep in your tokens object, and which ones you remove.

```
# 1 - Tokenize corpus & remove punctuation
hs_tokens <- tokens(hs_corpus,
                    remove_punct = TRUE,
                    remove_numbers = TRUE,
                    remove_symbols = TRUE,
                    remove_url = TRUE,
                    padding = TRUE) # For even more options, see ?tokens
head(hs_tokens[[7]], 20) # Gives me 20 tokens from the seventh document in corpus
```

```
##  [1] "Dwelling"    "is"          "a"           "fundamental" "factor"
##  [6] "for"         "mental"      "health"      ""            "Lockdowns"
## [11] ""            "established" "to"          "contain"     "the"
## [16] "spread"      "of"          "SARS-CoV-2"  ""            "forced"
```

```
# 2- Lowercase the corpus
hs_lower_tokens <- tokens_tolower(hs_tokens)
head(hs_lower_tokens[[7]], 20)
```

```
##  [1] "dwelling"    "is"          "a"           "fundamental" "factor"
##  [6] "for"         "mental"      "health"      ""            "lockdowns"
## [11] ""            "established" "to"          "contain"     "the"
## [16] "spread"      "of"          "sars-cov-2"  ""            "forced"
```

Your next choice is between discarding or not discarding words from the tokenized version of the corpus using a list of stopwords or by passing your own list of words. In either case, you will want to use the `tokens_remove()` command.

The `stopwords` package, which is used by `quanteda`, includes a good array of lists of commonly used words for many languages. The package includes lists from different sources, and for each source, there are lists for different languages. You can get the lists of sources and languages with specific commands as detailed below. Once you have identified the source and language you want, you can print the list of words.

```
# 3 - Remove stopwords
# Prints a list of available sources for stopwords
stopwords_getsources()
```

```
## [1] "snowball"      "stopwords-iso" "misc"          "smart"
## [5] "marimo"        "ancient"       "nltk"          "perseus"
```

```
# Prints a list of languags for a given source
stopwords_getlanguages("marimo")
```

```
## [1] "en"    "de"    "ru"    "ar"    "he"    "zh_tw" "zh_cn" "ko"    "ja"
```

```
stopwords("en", "nltk")
```

```
##   [1] "i"          "me"         "my"         "myself"     "we"
##   [6] "our"        "ours"       "ourselves"  "you"        "you're"
##  [11] "you've"     "you'll"     "you'd"      "your"       "yours"
##  [16] "yourself"   "yourselves" "he"         "him"        "his"
##  [21] "himself"    "she"        "she's"      "her"        "hers"
##  [26] "herself"    "it"         "it's"       "its"        "itself"
##  [31] "they"       "them"       "their"      "theirs"     "themselves"
##  [36] "what"       "which"      "who"        "whom"       "this"
##  [41] "that"       "that'll"    "these"      "those"      "am"
##  [46] "is"         "are"        "was"        "were"       "be"
##  [51] "been"       "being"      "have"       "has"        "had"
##  [56] "having"     "do"         "does"       "did"        "doing"
##  [61] "a"          "an"         "the"        "and"        "but"
##  [66] "if"         "or"         "because"    "as"         "until"
##  [71] "while"      "of"         "at"         "by"         "for"
##  [76] "with"       "about"      "against"    "between"    "into"
##  [81] "through"    "during"     "before"     "after"      "above"
##  [86] "below"      "to"         "from"       "up"         "down"
##  [91] "in"         "out"        "on"         "off"        "over"
##  [96] "under"      "again"      "further"    "then"       "once"
## [101] "here"       "there"      "when"       "where"      "why"
## [106] "how"        "all"        "any"        "both"       "each"
## [111] "few"        "more"       "most"       "other"      "some"
## [116] "such"       "no"         "nor"        "not"        "only"
## [121] "own"        "same"       "so"         "than"       "too"
## [126] "very"       "s"          "t"          "can"        "will"
## [131] "just"       "don"        "don't"      "should"     "should've"
## [136] "now"        "d"          "ll"         "m"          "o"
## [141] "re"         "ve"         "y"          "ain"        "aren"
## [146] "aren't"     "couldn"     "couldn't"   "didn"       "didn't"
## [151] "doesn"      "doesn't"    "hadn"       "hadn't"     "hasn"
## [156] "hasn't"     "haven"      "haven't"    "isn"        "isn't"
## [161] "ma"         "mightn"     "mightn't"   "mustn"      "mustn't"
## [166] "needn"      "needn't"    "shan"       "shan't"     "shouldn"
## [171] "shouldn't"  "wasn"       "wasn't"     "weren"      "weren't"
## [176] "won"        "won't"      "wouldn"     "wouldn't"
```

```
stopwords("en", "smart")
```

```
##   [1] "a"               "a's"             "able"            "about"
##   [5] "above"           "according"       "accordingly"     "across"
##   [9] "actually"        "after"           "afterwards"      "again"
##  [13] "against"         "ain't"           "all"             "allow"
##  [17] "allows"          "almost"          "alone"           "along"
##  [21] "already"         "also"            "although"        "always"
##  [25] "am"              "among"           "amongst"         "an"
##  [29] "and"             "another"         "any"             "anybody"
##  [33] "anyhow"          "anyone"          "anything"        "anyway"
##  [37] "anyways"         "anywhere"        "apart"           "appear"
##  [41] "appreciate"      "appropriate"     "are"             "aren't"
##  [45] "around"          "as"              "aside"           "ask"
##  [49] "asking"          "associated"      "at"              "available"
##  [53] "away"            "awfully"         "b"               "be"
##  [57] "became"          "because"         "become"          "becomes"
##  [61] "becoming"        "been"            "before"          "beforehand"
##  [65] "behind"          "being"           "believe"         "below"
##  [69] "beside"          "besides"         "best"            "better"
##  [73] "between"         "beyond"          "both"            "brief"
##  [77] "but"             "by"              "c"               "c'mon"
##  [81] "c's"             "came"            "can"             "can't"
##  [85] "cannot"          "cant"            "cause"           "causes"
##  [89] "certain"         "certainly"       "changes"         "clearly"
##  [93] "co"              "com"             "come"            "comes"
##  [97] "concerning"      "consequently"    "consider"        "considering"
## [101] "contain"         "containing"      "contains"        "corresponding"
## [105] "could"           "couldn't"        "course"          "currently"
## [109] "d"               "definitely"      "described"       "despite"
## [113] "did"             "didn't"          "different"       "do"
## [117] "does"          "doesn’t"       "doing"         "don’t"
## [121] "done"          "down"          "downwards"     "during"
## [125] "e"             "each"          "edu"           "eg"
## [129] "eight"         "either"        "else"          "elsewhere"
## [133] "enough"        "entirely"      "especially"    "et"
## [137] "etc"           "even"          "ever"          "every"
## [141] "everybody"     "everyone"      "everything"    "everywhere"
## [145] "ex"            "exactly"       "example"       "except"
## [149] "f"             "far"           "few"           "fifth"
## [153] "first"         "five"          "followed"      "following"
## [157] "follows"       "for"           "former"        "formerly"
## [161] "forth"         "four"          "from"          "further"
## [165] "furthermore"   "g"             "get"           "gets"
## [169] "getting"       "given"         "gives"         "go"
## [173] "goes"          "going"         "gone"          "got"
## [177] "gotten"        "greetings"     "h"             "had"
## [181] "hadn’t"        "happens"       "hardly"        "has"
## [185] "hasn’t"        "have"          "haven’t"       "having"
## [189] "he"            "he’s"          "hello"         "help"
## [193] "hence"         "her"           "here"          "here’s"
## [197] "hereafter"     "hereby"        "herein"        "hereupon"
## [201] "hers"          "herself"       "hi"            "him"
## [205] "himself"       "his"           "hither"        "hopefully"
## [209] "how"           "howbeit"       "however"       "i"
## [213] "i’d"           "i’ll"          "i’m"           "i’ve"
## [217] "ie"            "if"            "ignored"       "immediate"
## [221] "in"            "inasmuch"      "inc"           "indeed"
## [225] "indicate"      "indicated"     "indicates"     "inner"
## [229] "insofar"       "instead"       "into"          "inward"
## [233] "is"            "isn’t"         "it"            "it’d"
## [237] "it’ll"         "it’s"          "its"           "itself"
## [241] "j"             "just"          "k"             "keep"
## [245] "keeps"         "kept"          "know"          "knows"
## [249] "known"         "l"             "last"          "lately"
## [253] "later"         "latter"        "latterly"      "least"
## [257] "less"          "lest"          "let"           "let’s"
## [261] "like"          "liked"         "likely"        "little"
## [265] "look"          "looking"       "looks"         "ltd"
## [269] "m"             "mainly"        "many"          "may"
## [273] "maybe"         "me"            "mean"          "meanwhile"
## [277] "merely"        "might"         "more"          "moreover"
## [281] "most"          "mostly"        "much"          "must"
## [285] "my"            "myself"        "n"             "name"
## [289] "namely"        "nd"            "near"          "nearly"
## [293] "necessary"     "need"          "needs"         "neither"
## [297] "never"         "nevertheless"  "new"           "next"
## [301] "nine"          "no"            "nobody"        "non"
## [305] "none"          "noone"         "nor"           "normally"
## [309] "not"           "nothing"       "novel"         "now"
## [313] "nowhere"       "o"             "obviously"     "of"
## [317] "off"           "often"         "oh"            "ok"
## [321] "okay"          "old"           "on"            "once"
## [325] "one"           "ones"          "only"          "onto"
## [329] "or"            "other"         "others"        "otherwise"
## [333] "ought"         "our"           "ours"          "ourselves"
## [337] "out"           "outside"       "over"          "overall"
## [341] "own"           "p"             "particular"    "particularly"
## [345] "per"           "perhaps"       "placed"        "please"
## [349] "plus"          "possible"      "presumably"    "probably"
## [353] "provides"      "q"             "que"           "quite"
## [357] "qv"            "r"             "rather"        "rd"
## [361] "re"            "really"        "reasonably"    "regarding"
## [365] "regardless"    "regards"       "relatively"    "respectively"
## [369] "right"         "s"             "said"          "same"
## [373] "saw"           "say"           "saying"        "says"
## [377] "second"        "secondly"      "see"           "seeing"
## [381] "seem"          "seemed"        "seeming"       "seems"
## [385] "seen"          "self"          "selves"        "sensible"
## [389] "sent"          "serious"       "seriously"     "seven"
## [393] "several"       "shall"         "she"           "should"
## [397] "shouldn't"     "since"         "six"           "so"
## [401] "some"          "somebody"      "somehow"       "someone"
## [405] "something"     "sometime"      "sometimes"     "somewhat"
## [409] "somewhere"     "soon"          "sorry"         "specified"
## [413] "specify"       "specifying"    "still"         "sub"
## [417] "such"          "sup"           "sure"          "t"
## [421] "t's"           "take"          "taken"         "tell"
## [425] "tends"         "th"            "than"          "thank"
## [429] "thanks"        "thanx"         "that"          "that's"
## [433] "thats"         "the"           "their"         "theirs"
## [437] "them"          "themselves"    "then"          "thence"
## [441] "there"         "there's"       "thereafter"    "thereby"
## [445] "therefore"     "therein"       "theres"        "thereupon"
## [449] "these"         "they"          "they'd"        "they'll"
## [453] "they're"       "they've"       "think"         "third"
## [457] "this"          "thorough"      "thoroughly"    "those"
## [461] "though"        "three"         "through"       "throughout"
## [465] "thru"          "thus"          "to"            "together"
## [469] "too"           "took"          "toward"        "towards"
## [473] "tried"         "tries"         "truly"         "try"
## [477] "trying"        "twice"         "two"           "u"
## [481] "un"            "under"         "unfortunately" "unless"
## [485] "unlikely"      "until"         "unto"          "up"
## [489] "upon"          "us"            "use"           "used"
## [493] "useful"        "uses"          "using"         "usually"
## [497] "uucp"          "v"             "value"         "various"
## [501] "very"          "via"           "viz"           "vs"
## [505] "w"             "want"          "wants"         "was"
## [509] "wasn't"        "way"           "we"            "we'd"
## [513] "we'll"         "we're"         "we've"         "welcome"
## [517] "well"          "went"          "were"          "weren't"
## [521] "what"          "what's"        "whatever"      "when"
## [525] "whence"        "whenever"      "where"         "where's"
## [529] "whereafter"    "whereas"       "whereby"       "wherein"
## [533] "whereupon"     "wherever"      "whether"       "which"
## [537] "while"         "whither"       "who"           "who's"
## [541] "whoever"       "whole"         "whom"          "whose"
## [545] "why"           "will"          "willing"       "wish"
## [549] "with"           "within"         "without"        "won't"
## [553] "wonder"         "would"          "would"          "wouldn't"
## [557] "x"              "y"              "yes"            "yet"
## [561] "you"            "you'd"          "you'll"         "you're"
## [565] "you've"         "your"           "yours"          "yourself"
## [569] "yourselves"     "z"              "zero"
```

Keep in mind that, depending on the list of stopwords that you decide to use to remove commonly occurring features in the corpus, the size of your vocabulary will change significantly. Some projects might require you to keep words like "zero" or "willing", which are not in the "nltk - en" list of stopwords, but they are in the "smart - en" list. So, in that case, you'd want to use one list of stopwords and not the other. However, this is not set in stone, and different projects will call for different approaches.

```r
# Exclude words from stopwords list "nltk - en"
hs_tokens_no_stopwords <- tokens_remove(hs_lower_tokens,
                                        stopwords("en", "nltk"))
head(hs_lower_tokens[[7]], 20) # with stopwords
```

```
##  [1] "dwelling"    "is"          "a"           "fundamental" "factor"
##  [6] "for"         "mental"      "health"      ""            "lockdowns"
## [11] ""            "established" "to"          "contain"     "the"
## [16] "spread"      "of"          "sars-cov-2"  ""            "forced"
```

```r
head(hs_tokens_no_stopwords[[7]], 20) # without stopwords
```

```
##  [1] "dwelling"    "fundamental" "factor"      "mental"      "health"
##  [6] ""            "lockdowns"   ""            "established" "contain"
## [11] "spread"      "sars-cov-2"  ""            "forced"      "millions"
## [16] "people"      "take"        "shelter"     "homes"       ""
```

In addition, if you believe this could help your analysis, you could also create your own list of words by simply creating a vector of words (or importing a list from an external file), and removing all those from your tokens object. In our example, because many of the Abstracts mention the words "Taylor Francis Group", and this is not information that we will find particularly useful in any analysis, we might want to remove these three words at this stage.

```r
# Create a list of words to exclude
words_to_exclude <- c("taylor", "francis", "group")

# Exclude words from custom made list
hs_tokens_no_stopwords <- tokens_remove(hs_tokens_no_stopwords, words_to_exclude)
```

The final step of the pre-processing stage involves stemming or lemmatizing your corpus. Both approaches reduce the size of our data, as words that would be considered different in an un-stemmed corpus (e.g. win, winner and winning), would become the same word. Stemming can be done fairly quickly, but it is more prone to error. Lemmatizinig is more computationally intensive, but much more accurate.

We can use the `tokens_wordstem()` command to stem a sentence, a text, or a `quanteda` corpus. By default, quanteda assumes we are stemming an English language text, but it is possible to use the argument `language` to specify an alternative language.

```r
# 4 - Stemming your tokens object
hs_tokens_stemmed <- tokens_wordstem(hs_tokens_no_stopwords)
head(hs_tokens_stemmed[[7]], 20)
```

```
##  [1] "dwell"      "fundament"  "factor"     "mental"     "health"
##  [6] ""           "lockdown"   ""           "establish"  "contain"
## [11] "spread"     "sars-cov-2" ""           "forc"       "million"
## [16] "peopl"      "take"       "shelter"    "home"       ""
```

Lemmatizing involves using previously trained models of a language that make it possible to identify what part of speech a given word is, or to disambiguate when a word might have different meanings. In this process, lemmatizing (reducing the words to it's root or lemma) is a lot more precise. This is, as you might imagine, a much more computationally intensive process than stemming, which we were able to complete rather fast. There's no function in `quanteda` to lemmatize a corpus, but we can lean on the `udpipe` package to do so.

As this is a rather more elaborate process, we are not going to cover it today. You may want to refer to the udpipe package online documentation for examples and processes.

There is a trade-off between speed and efficiency, and accuracy in the choice between stemming and lemmatizing. This decision, as well as the previous ones we took during the pre-processing stage will have an effect on the size of the document feature matrix, the data structure we will create next as we prepare to do some analysis of our data.

Now that we have completed the pre-processing of our data, we are ready to create our first DFM or Document Feature Matrix. This is the data structure used to fit statistical models for text analysis in `quanteda`. This is just one way to represent data in the bag-of-words-approach. Let's first use the `dfm()` function or command to create a DFM from the stemmed tokens object that we saved early on.

```r
# DFM fromm a lemmatized tokens object
hs_dfm_stemmed <- dfm(hs_tokens_stemmed)
hs_dfm_stemmed
```

```
## Document-feature matrix of: 1,422 documents, 7,741 features (99.08% sparse) and 26 docvars.
##                        features
## docs                       articl examin intern literatur placemaking-practic
##   housing_scopus.csv.1 24       2      1      1         2                   1
##   housing_scopus.csv.2 28       0      0      0         0                   0
##   housing_scopus.csv.3 18       0      1      0         0                   0
##   housing_scopus.csv.4 18       0      0      0         1                   0
##   housing_scopus.csv.5 24       0      1      0         0                   0
##   housing_scopus.csv.6 27       1      0      1         0                   0
##                        features
## docs                     initi encourag sens place-in
##   housing_scopus.csv.1     1        1    1        1
##   housing_scopus.csv.2     0        0    0        0
##   housing_scopus.csv.3     0        0    0        0
##   housing_scopus.csv.4     0        0    0        0
##   housing_scopus.csv.5     0        0    1        0
##   housing_scopus.csv.6     0        0    0        0
## [ reached max_ndoc ... 1,416 more documents, reached max_nfeat ... 7,731 more features ]
```

```r
ndoc(hs_dfm_stemmed) # Gives us the number of documents
```

```
## [1] 1422
```

```r
nfeat(hs_dfm_stemmed) # Gives us the number of (unique) features
```

```
## [1] 7741
```

So we have reached the end of our pre-processing stage. We now have a DFM, which we could use to fit a range of statistical models for analysis. We will see that in a minute when we apply a dictionary to our dataset. Before we do that, let's have a look at our DFM, and think about some possible "analyses".

With a DFM, it is very easy to get the most frequent words in our corpus using the `topfeatures()` command.

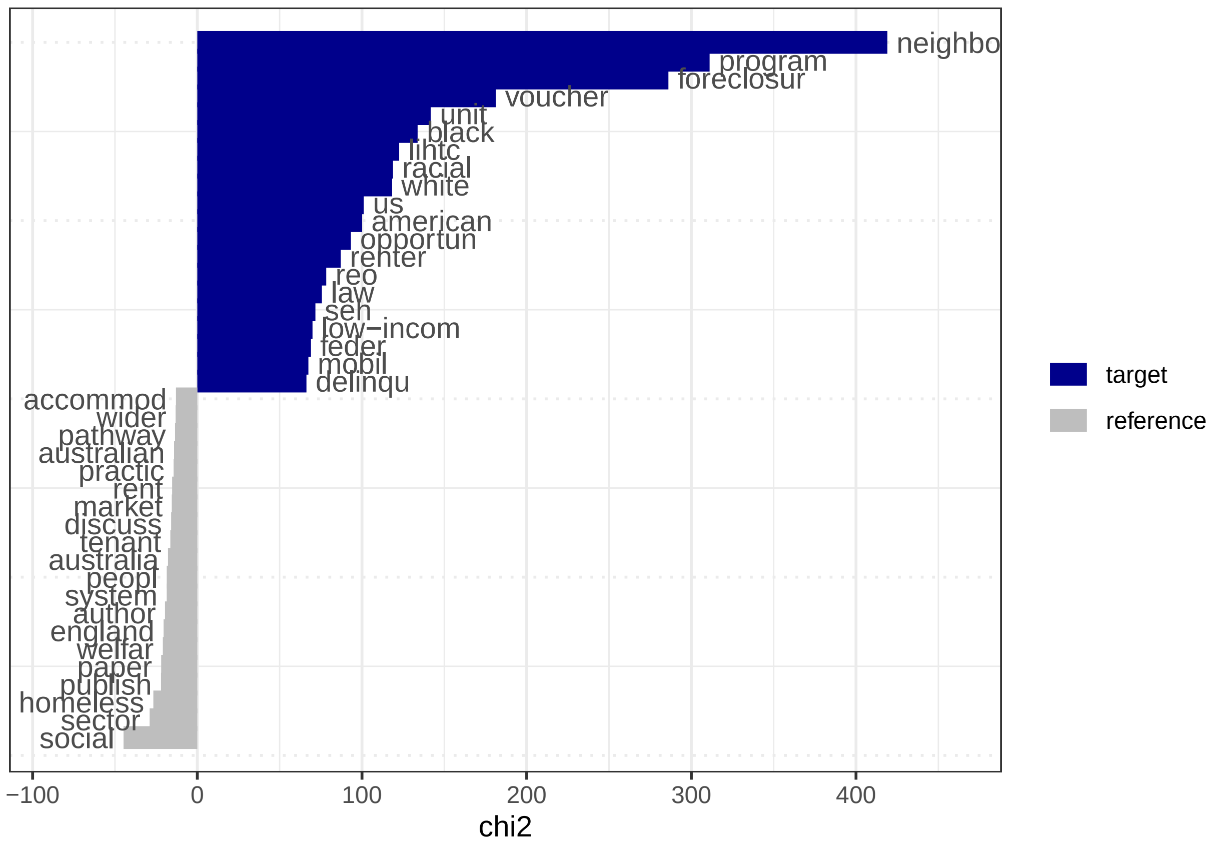
```r
# Most frequently occurring words
topfeatures(hs_dfm_stemmed, 20)
```

```
##                hous     paper    social    polici     studi       use        uk 
##     30333      5103      1279      1191      1144       946       863       809 
##    market     limit     trade   informa   develop household      home     chang 
##       802       774       636       627       619       596       595       560 
##    examin    effect      find    differ 
##       531       519       518       507
```

This, in itself, does not tell us much. Given that our data comes from a journal on housing studies, there's is little to gain from knowing that the word "hous" is the most common. However, there's some other things we could do. For example, we could compare the most frequently used words in abstracts written by US-based scholars, and compare them to those that are not based in the US. Remember, we have some of this information in our corpus metadata.

A common way to compare word use across groups is a measure called "keyness", which is nothing but a score for features that occur differentially across different categories. This is widely used in disciplines like corpus linguistics. We will first compute "keyness" for our two groups, and we will then plot the data for easier interpretation

```r
# We compute keyness using the variable united_states
# as the grouping variable
tstat_key <- textstat_keyness(hs_dfm_stemmed,
                              docvars(hs_dfm_stemmed, "united_states") > 0)
textplot_keyness(tstat_key)
```



```r
head(tstat_key, 10) # Words strongly associated with US-based scholars
```

```
##         feature     chi2 p n_target n_reference
## 1  neighborhood 419.0566 0      111          55
## 2       program 311.0964 0       95          61
## 3     foreclosur 286.0687 0       58          14
## 4       voucher 181.3249 0       31           2
## 5          unit 141.7299 0       63          67
## 6         black 133.7768 0       30          10
## 7         lihtc 122.5815 0       22           2
## 8        racial 118.8442 0       29          12
## 9         white 118.3148 0       31          15
## 10           us 101.0423 0       51          62
```

```r
tail(tstat_key, 10) # Words strongly associated with non US-based scholars
```

```
##       feature      chi2            p n_target n_reference
## 7732    peopl -18.56164 1.644973e-05       23         371
## 7733   system -18.57863 1.630378e-05       16         303
## 7734   author -19.62163 9.439471e-06        8         227
## 7735  england -20.48984 5.994852e-06        0         135
## 7736   welfar -20.93681 4.746853e-06        7         225
## 7737    paper -21.93781 2.816301e-06      112        1167
## 7738  publish -22.03711 2.674295e-06        1         159
## 7739 homeless -26.69568 2.381539e-07       26         465
## 7740   sector -28.96977 7.351652e-08       11         326
## 7741   social -44.80840 2.172884e-11       79        1112
```

What can be seen in the plot is that the target group (abstracts from authors based in the US) use words such as “foreclosure”, “voucher” or “black” significantly more often than abstracts from authors not based in the US. To plot this figure, `quanteda` relies on the data that you can see in the tables.

This is just one of multiple approaches we could use to get started in our data analysis. This should be driven by your research questions and/or hypotheses. You can have a look at the different plots and statistical measures (for example, text similarity or language complexity) available through `quanteda` on their website.

With this one example, we conclude Demo 2, which focused on pre-processing on `quanteda`. As we discussed in the lecture, there are usually seven (some might say 8) steps in any computational text analysis project:

1. Selecting texts and defining a corpus. [Demo 1]
2. Converting the texts into a common format. [Demo 1]
3. Deciding the documentary unit. [Demo 1]
4. Defining and refining the features. [Demo 2]
5. Converting features to a quantitative matrix. [Demo 2]
6. Extracting information from the matrix statistically. [Demo 2]
7. Summarizing & interpreting the results. [Demo 2]
8. Validating the results. [No time for this today, but super important]

Demo 3 - Applying a dictionary using `quanteda`

Once you have pre-processed your text and created a structured dataset of your corpus, in our case in the form of a DFM, a range of possibilities open up in terms of analysis. Once your data is in a DFM format, you are able to use a range of unsupervised machine learning methods for topic classification (for example, STM or structural topic model, which will be discussed in Text Analysis 2).

For simplicity, in this final demo, we will use a dictionary, which is a relatively simple form of supervised classification of documents. The principle of dictionaries in computational text analysis is that, by counting the frequency of certain words, we might be able to classify documents into different categories. Once we have classified our documents, we might be able to answer certain research questions.

To guide us through this final demonstration, we are going to work with one research question:

- RQ1: What are the most often discussed countries in abstracts of papers published in *Housing Studies*?

The process of applying a dictionary is relatively straightforward:

1. Create a corpus
2. Tokenise the corpus and pre-process texts
3. Identify a relevant dictionary (or create one)
4. Apply the dictionary to your dataset to count frequencies of words
5. Summarise the findings and further analysis

6. Validate, validate, validate.

```r
# 1 - Create a corpus
# 2 - Tokenise the corpus & pre-process texts
# 3 - Identify a dictionary
dict_countries <- data_dictionary_newsmap_en
dict_countries
```

```
## Dictionary object with 5 primary key entries and 3 nested levels.
## - [AFRICA]:
##   - [EAST]:
##     - [BI]:
##       - burundi, burundian*, bujumbura
##     - [DJ]:
##       - djibouti, djiboutian*
##     - [ER]:
##       - eritrea, eritrean*, asmara
##     - [ET]:
##       - ethiopia, ethiopian*, addis ababa
##     - [KE]:
##       - kenya, kenyan*, nairobi
##     - [KM]:
##       - comoros, comorian*, moroni
##     [ reached max_nkey ... 13 more keys ]
##   - [MIDDLE]:
##     - [AO]:
##       - angola, angolan*, luanda
##     - [CD]:
##       - democratic republic congo, dr congo, drc, democratic republic congolese, dr congolese, kinsha
##     - [CF]:
##       - central african republic, central african*, bangui
##     - [CG]:
##       - congo, congo republic, congolese, brazzaville
##     - [CM]:
##       - cameroon, cameroonian*, yaounde, yaoundé
##     - [GA]:
##       - gabon, gabonese, libreville
##     [ reached max_nkey ... 3 more keys ]
##   - [NORTH]:
##     - [DZ]:
##       - algeria, algerian*, algiers
##     - [EG]:
##       - egypt, egyptian*, cairo
##     - [EH]:
##       - western sahara, western saharan*, el aaiun
##     - [LY]:
##       - libya, libyan*, tripoli
##     - [MA]:
##       - morocco, moroccan*, rabat
##     - [SD]:
##       - sudan, sudanese, khartoum
##     [ reached max_nkey ... 2 more keys ]
##   - [SOUTH]:
##     - [BW]:
```

```
##       - botswana, botswanan*, gaborone
##     - [LS]:
##       - lesotho, lesothonian*, maseru
##     - [NA]:
##       - namibia, namibian*, windhoek
##     - [SZ]:
##       - swaziland, swazi*, lobamba, mbabane
##     - [ZA]:
##       - south africa, s african, sa, south african*, s african*, cape town, johannesburg, pretoria
##   - [WEST]:
##     - [BF]:
##       - burkina faso, burkinabe*, ouagadougou
##     - [BJ]:
##       - benin, beninese, beninois, porto novo
##     - [CI]:
##       - ivory coast, côte d'ivoire, i coast, ivorian*, yamoussoukro, abidjan
##     - [CV]:
##       - cape verde, cape verdean*, praia
##     - [GH]:
##       - ghana, ghanaian*, accra
##     - [GM]:
##       - gambia, gambian*, banjul
##     [ reached max_nkey ... 11 more keys ]
## - [AMERICA]:
##   - [CARIB]:
##     - [AG]:
##       - antigua and barbuda, antiguan*, barbudan*
##     - [AI]:
##       - anguilla, anguillan*, the valley
##     - [AW]:
##       - aruba, aruban*, oranjestad
##     - [BB]:
##       - barbados, barbadian*, bridgetown
##     - [BL]:
##       - saint barthelemy, saint-barthelemy, saint-barthélemy, st barthelemy, barthelemois, gustavia
##     - [BQ]:
##       - bonaire, bonairean*, kralendijk
##     [ reached max_nkey ... 22 more keys ]
##   - [CENTER]:
##     - [BZ]:
##       - belize, belizean*, belmopan
##     - [CR]:
##       - costa rica, costa rican*, ticos, san jose
##     - [GT]:
##       - guatemala, guatemalan*, guatemala city
##     - [HN]:
##       - honduras, honduran*, tegucigalpa
##     - [MX]:
##       - mexico, mexican*, mexico city
##     - [NI]:
##       - nicaragua, nicaraguan*, managua
##     [ reached max_nkey ... 2 more keys ]
##   - [SOUTH]:
##     - [AR]:
```

```
##       - argentina, argentine*, argentinian*, buenos aires
##     - [BO]:
##       - bolivia, bolivian*, sucre, la paz
##     - [BR]:
##       - brazil, brazilian*, brasilia, sao paulo, rio
##     - [CL]:
##       - chile, chilean*, santiago
##     - [CO]:
##       - colombia, colombian*, bogota
##     - [EC]:
##       - ecuador, ecuadorian*, quito
##     [ reached max_nkey ... 8 more keys ]
##   - [NORTH]:
##     - [BM]:
##       - bermuda, bermudan*
##     - [CA]:
##       - canada, canadian*, ottawa, toronto, quebec
##     - [GL]:
##       - greenland, greenlander*, nuuk
##     - [PM]:
##       - saint pierre and miquelon, st pierre and miquelon, saint pierrais, miquelonnais, saint pierr
##     - [US]:
##       - united states, us, american*, washington, new york
## - [ASIA]:
##   - [CENTER]:
##     - [KG]:
##       - kyrgyzstan, kyrgyz*, bishkek
##     - [KZ]:
##       - kazakhstan, kazakh*, astana
##     - [TJ]:
##       - tajikistan, tajiks*, dushanbe
##     - [TM]:
##       - turkmenistan, turkmen*, ashhabad
##     - [UZ]:
##       - uzbekistan, uzbek*, tashkent
##   - [EAST]:
##     - [CN]:
##       - china, chinese, beijing, shanghai
##     - [HK]:
##       - hong kong, hongkongese
##     - [JP]:
##       - japan, japanese, tokyo
##     - [KP]:
##       - north korea, n korea, north korean*, n korean*, dprk, pyongyang
##     - [KR]:
##       - south korea, s korea, south korean, s korean*, seoul
##     - [MN]:
##       - mongolia, mongolian*, ulan bator
##     [ reached max_nkey ... 2 more keys ]
##   - [SOUTH]:
##     - [AF]:
##       - afghanistan, afghan*, kabul
##     - [BD]:
##       - bangladesh, bangladeshi*, dhaka, dacca
##     - [BT]:
##       - bhutan, bhutanese, thimphu
##     - [IN]:
##       - india, indian*, mumbai, new delhi
##     - [IR]:
##       - iran, iranian*, tehran
##     - [LK]:
##       - sri lanka, sri lankan*, colombo
##     [ reached max_nkey ... 3 more keys ]
##   - [SOUTH-EAST]:
##     - [BN]:
##       - brunei, bruneian*
##     - [ID]:
##       - indonesia, indonesian*, jakarta
##     - [KH]:
##       - cambodia, cambodian*, phnom penh
##     - [LA]:
##       - laos, laotian*, vientiane
##     - [MM]:
##       - myanmar, burma, myanmarese, burmese, yangon, naypyidaw
##     - [MY]:
##       - malaysia, malaysian*, kuala lumpur, putrajaya
##     [ reached max_nkey ... 5 more keys ]
##   - [WEST]:
##     - [AE]:
##       - united arab emirates, uae, emirati*, emiri*, dubai, abu dhabi
##     - [AM]:
##       - armenia, armenian*, yerevan
##     - [AZ]:
##       - azerbaijan, azerbaijani*, azeri*, baku
##     - [BH]:
##       - bahrain, bahraini*, manama
##     - [CY]:
##       - cyprus, cypriot*, nicosia
##     - [GE]:
##       - georgia, georgian*, tbilisi
##     [ reached max_nkey ... 12 more keys ]
## - [EUROPE]:
##   - [EAST]:
##     - [BG]:
##       - bulgaria, bulgarian*, sofia
##     - [BY]:
##       - belarus, belarusian*, minsk
##     - [CZ]:
##       - czech republic, czech*, prague
##     - [HU]:
##       - hungary, hungarian*, budapest
##     - [MD]:
##       - moldova, moldovan*, chisinau
##     - [PL]:
##       - poland, polish, pole*, warsaw
##     [ reached max_nkey ... 4 more keys ]
##   - [NORTH]:
##     - [AX]:
```

```
##       - aland islands, aland island*, alandish, mariehamn
##     - [DK]:
##       - denmark, danish, dane*, copenhagen
##     - [EE]:
##       - estonia, estonian*, tallinn
##     - [FI]:
##       - finland, finnish, finn*, helsinki
##     - [FO]:
##       - faeroe islands, faeroe island*, faroese*, torshavn
##     - [GB]:
##       - uk, united kingdom, britain, british, briton*, brit*, london
##     [ reached max_nkey ... 10 more keys ]
##   - [SOUTH]:
##     - [AD]:
##       - andorra, andorran*
##     - [AL]:
##       - albania, albanian*, tirana
##     - [BA]:
##       - bosnia, bosnian*, bosnia and herzegovina, herzegovina, sarajevo
##     - [ES]:
##       - spain, spanish, spaniard*, madrid, barcelona
##     - [GI]:
##       - gibraltar, gibraltarian*, llanitos
##     - [GR]:
##       - greece, greek*, athens
##     [ reached max_nkey ... 11 more keys ]
##   - [WEST]:
##     - [AT]:
##       - austria, austrian*, vienna
##     - [BE]:
##       - belgium, belgian*, brussels
##     - [CH]:
##       - switzerland, swiss*, zurich, bern
##     - [DE]:
##       - germany, german*, berlin, frankfurt
##     - [FR]:
##       - france, french*, paris
##     - [LI]:
##       - liechtenstein, liechtenstein*, vaduz
##     [ reached max_nkey ... 3 more keys ]
## - [OCEANIA]:
##   - [AU-NZ]:
##     - [AU]:
##       - australia, australian*, aussie*, oz, canberra, sydney
##     - [CK]:
##       - cook islands, cook island*, avarua
##     - [NF]:
##       - norfolk island, norfolk islander*
##     - [NZ]:
##       - new zealand, n zealand, nz, new zealander*, kiwi*, wellington, auckland
##   - [MEL]:
##     - [FJ]:
##       - fiji, fijian*
##     - [NC]:
```

```
##       - new caledonia, new caledonian*, noumea
##     - [PG]:
##       - papua new guinea, papua new guinean*, papuan*, port moresby
##     - [SB]:
##       - solomon islands, solomon island*, honiara
##     - [VU]:
##       - vanuatu, vanuatuan*, port vila
##   - [MIC]:
##     - [FM]:
##       - micronesia, micronesian*, palikir
##     - [GU]:
##       - guam, guamanian*, hagatna
##     - [KI]:
##       - kiribati, kiribati*, tarawa
##     - [MH]:
##       - marshall islands, marshall island*, marshallese, majuro
##     - [MP]:
##       - northern mariana islands, northern mariana island*, capital hill
##     - [NR]:
##       - nauru, nauruan*, yaren
##     [ reached max_nkey ... 1 more key ]
##   - [POL]:
##     - [AS]:
##       - american samoa, american samoan*, pago pago
##     - [NU]:
##       - niue, niuean*, alofi
##     - [PF]:
##       - french polynesia, french polynesian*, papeete
##     - [PN]:
##       - pitcairn islands, pitcairn island*, adamstown
##     - [TK]:
##       - tokelau, tokelauan*, nukunonu
##     - [TO]:
##       - tonga, tongan*, nuku'alofa
##     [ reached max_nkey ... 3 more keys ]
```

```
# 4 - Apply the dictionary to the dataset
countries_abstract <- tokens_lookup(hs_tokens,
                                    dictionary = dict_countries,
                                    levels = 3)

# 5 - Summarise the instances
dfm_countries <- dfm(countries_abstract)
dfm_countries
```

```
## Document-feature matrix of: 1,422 documents, 241 features (99.48% sparse) and 26 docvars.
##                        features
## docs                     bi dj er et ke km mg mu mw mz
##   housing_scopus.csv.1  0  0  0  0  0  0  0  0  0  0
##   housing_scopus.csv.2  0  0  0  0  0  0  0  0  0  0
##   housing_scopus.csv.3  0  0  0  0  0  0  0  0  0  0
##   housing_scopus.csv.4  0  0  0  0  0  0  0  0  0  0
##   housing_scopus.csv.5  0  0  0  0  0  0  0  0  0  0
##   housing_scopus.csv.6  0  0  0  0  0  0  0  0  0  0
## [ reached max_ndoc ... 1,416 more documents, reached max_nfeat ... 231 more features ]
```

```
# The function topfeatures can summarise the data for us
topfeatures(dfm_countries, 10)
```

```
##  gb  au  us  cn  nl  ca  ie  se  de  nz
## 925 253 211 184 144  97  91  74  64  45
```

```
# We can easily compare groups
topfeatures(dfm_countries, 10, groups = united_states)
```

```
## $`0`
##  gb  au  cn  nl  us  ca  ie  se  de  nz
## 844 252 151 142 101  96  91  73  62  43
##
## $`1`
##  us  gb  cn  tw  in  gh  ht  lt  ke  za
## 110  81  33   9   5   4   4   3   2   2
```

The findings that we find above appear to be quite "accurate". After all, it is difficult that a dictionary of countries and cities would go wrong. However, in this case, as in any other case, you'd want to add one last step to your analysis to validate the accuracy of your findings. This could be done, for example, by having a human code a small percentage of the data (say 10%), and then you could compute a measure of inter-coder reliability. Validation is ALWAYS a necessary final step in any computational text analysis project.

To answer RQ1, the information we get using `topfeatures` above would probably be enough. However, you might want to add some nuance and do some cross-tabs (i.e., comparisons between groups). For that, it might be helpful to merge our newly created variables, one for each country in the world, to our main dataset, which was called `df_abstracts`.

As DFMs are nothing by matrices, it is easy to convert them to dataframes and other forms of tabular data with the command `convert`. Once we have country mentions in a data.frame format, the merging could be done quite easily with a tidyverse verb like `left_join`, using the column "doc_id", which is in both our original dataset and in the newly created data.frame of country names, as the index for merging the data.

```
# We transform the DFM of country mentions to a data.frame format
df_countries <- convert(dfm_countries, to = "data.frame")

# We merge this information with our original dataset
df_abstracts %<>%
  left_join(df_countries, by = "doc_id")
```

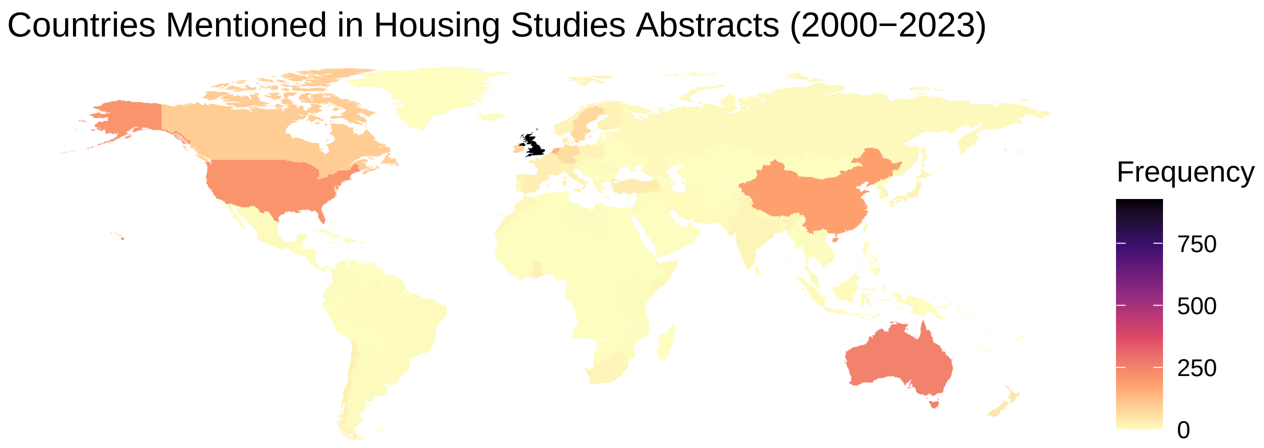
With this newly joined dataset, the opportunities for analysis are quite open. You could use some of the other variables in our dataset such as year or quant (which tells us whether an Abstract uses quantitative methods) to run further comparative analyses.

```
# Bonus Points: We could plot a heatmap to visualise our findings
country_counts <- as.data.frame(topfeatures(dfm_countries, 242))
country_counts$id <- toupper(rownames(country_counts))
colnames(country_counts) <- c("frequency", "id")
rownames(country_counts) <- NULL
country_counts
```

```
##    frequency id
## 1        925 GB
## 2        253 AU
## 3        211 US
## 4        184 CN
## 5        144 NL
## 6         97 CA
## 7         91 IE
## 8         74 SE
## 9         64 DE
## 10        45 NZ
## 11        38 HK
## 12        34 TR
## 13        30 FR
## 14        26 DK
## 15        26 ES
## 16        26 IT
## 17        25 GH
## 18        24 CL
## 19        24 FI
## 20        24 BE
## 21        22 PL
## 22        19 NO
## 23        18 IN
## 24        17 ZA
## 25        17 JP
## 26        16 TW
## 27        16 CH
## 28        15 SG
## 29        14 IL
## 30        13 CZ
## 31        12 MA
## 32        11 SO
## 33        10 KR
## 34        10 AT
## 35         9 RU
## 36         8 BD
## 37         8 MY
## 38         8 PS
## 39         8 PT
## 40         7 GR
## 41         6 BR
## 42         6 IR
## 43         6 PK
## 44         6 ID
## 45         6 AL
## 46         6 SI
## 47         5 AR
## 48         5 HU
## 49         4 UG
## 50         4 HT
## 51         4 MX
## 52         4 TH
## 53         4 VN
## 54           3 CO
## 55           3 MO
## 56           3 NP
## 57           3 RO
## 58           3 LT
## 59           2 KE
## 60           2 MZ
## 61           2 UY
## 62           2 JO
## 63           2 LB
## 64           2 IS
## 65           2 MT
## 66           2 LU
## 67           1 RW
## 68           1 TZ
## 69           1 LY
## 70           1 PR
## 71           1 CR
## 72           1 HN
## 73           1 NI
## 74           1 SV
## 75           1 EC
## 76           1 SR
## 77           1 PH
## 78           1 AM
## 79           1 SY
## 80           0 BI
## 81           0 DJ
## 82           0 ER
## 83           0 ET
## 84           0 KM
## 85           0 MG
## 86           0 MU
## 87           0 MW
## 88           0 RE
## 89           0 SC
## 90           0 YT
## 91           0 ZM
## 92           0 ZW
## 93           0 AO
## 94           0 CD
## 95           0 CF
## 96           0 CG
## 97           0 CM
## 98           0 GA
## 99           0 GQ
## 100          0 ST
## 101          0 TD
## 102          0 DZ
## 103          0 EG
## 104          0 EH
## 105          0 SD
## 106          0 SS
## 107          0 TN
## 108           0 BW
## 109           0 LS
## 110           0 NA
## 111           0 SZ
## 112           0 BF
## 113           0 BJ
## 114           0 CI
## 115           0 CV
## 116           0 GM
## 117           0 GN
## 118           0 GW
## 119           0 LR
## 120           0 ML
## 121           0 MR
## 122           0 NE
## 123           0 NG
## 124           0 SH
## 125           0 SL
## 126           0 SN
## 127           0 TG
## 128           0 AG
## 129           0 AI
## 130           0 AW
## 131           0 BB
## 132           0 BL
## 133           0 BQ
## 134           0 BS
## 135           0 CU
## 136           0 CW
## 137           0 DM
## 138           0 DO
## 139           0 GD
## 140           0 GP
## 141           0 JM
## 142           0 KN
## 143           0 KY
## 144           0 LC
## 145           0 MF
## 146           0 MQ
## 147           0 MS
## 148           0 SX
## 149           0 TC
## 150           0 TT
## 151           0 VC
## 152           0 VG
## 153           0 VI
## 154           0 BZ
## 155           0 GT
## 156           0 PA
## 157           0 BO
## 158           0 FK
## 159           0 GF
## 160           0 GY
## 161           0 PE
## 162          0 PY
## 163          0 VE
## 164          0 BM
## 165          0 GL
## 166          0 PM
## 167          0 KG
## 168          0 KZ
## 169          0 TJ
## 170          0 TM
## 171          0 UZ
## 172          0 KP
## 173          0 MN
## 174          0 AF
## 175          0 BT
## 176          0 LK
## 177          0 MV
## 178          0 BN
## 179          0 KH
## 180          0 LA
## 181          0 MM
## 182          0 TL
## 183          0 AE
## 184          0 AZ
## 185          0 BH
## 186          0 CY
## 187          0 GE
## 188          0 IQ
## 189          0 KW
## 190          0 OM
## 191          0 QA
## 192          0 SA
## 193          0 YE
## 194          0 BG
## 195          0 BY
## 196          0 MD
## 197          0 SK
## 198          0 UA
## 199          0 AX
## 200          0 EE
## 201          0 FO
## 202          0 GG
## 203          0 IM
## 204          0 JE
## 205          0 LV
## 206          0 SJ
## 207          0 AD
## 208          0 BA
## 209          0 GI
## 210          0 HR
## 211          0 KV
## 212          0 ME
## 213          0 MK
## 214          0 RS
## 215          0 SM
## 216          0 VA
## 217          0 LI
## 218          0 MC
## 219          0 CK
## 220          0 NF
## 221          0 FJ
## 222          0 NC
## 223          0 PG
## 224          0 SB
## 225          0 VU
## 226          0 FM
## 227          0 GU
## 228          0 KI
## 229          0 MH
## 230          0 MP
## 231          0 NR
## 232          0 PW
## 233          0 AS
## 234          0 NU
## 235          0 PF
## 236          0 PN
## 237          0 TK
## 238          0 TO
## 239          0 TV
## 240          0 WF
## 241          0 WS
```

```
# Next, we need to import geographic data to plot the map
world_map <- map_data(map = "world")
world_map$region <- iso.alpha(world_map$region) # convert contry name to ISO code

# We can plot the frequency of countries utih the geom_map function
ggplot(country_counts, aes(map_id = id)) +
  geom_map(aes(fill = frequency), map = world_map) +
  expand_limits(x = world_map$long, y = world_map$lat) +
  scale_fill_viridis_c(option = "A",
                       name = "Frequency",
                       direction = -1) +
  theme_void() +
  coord_fixed() +
  ggtitle("Countries Mentioned in Housing Studies Abstracts (2000-2023)")
```


Countries Mentioned in Housing Studies Abstracts (2000−2023)
Frequency
750
500
250
0