Introduction to Quantitative Text Analysis (QTA)

Short Course: Day One Lab

Dr. Brian J. Fogarty

8 May 2018

1. Set working directory, library, and install packages

This is about just getting set-up. More details at https://tutorials.quanteda.io/introduction/install/. Because I often have problems with my R library when connected with Glasgow's network, I set my library to my local R library using `.libPaths()`.

```
setwd("C:/QTA/Day One")
getwd()
```

```
## [1] "C:/QTA/Day One"
```

```
.libPaths("C:/Program Files/R/R-3.4.2/library")
.libPaths()
```

```
## [1] "C:/Program Files/R/R-3.4.2/library"
```

```
install.packages("quanteda", dependencies = TRUE)
install.packages("readtext")
install.packages("devtools")
devtools::install_github("quanteda/quanteda.corpora")
```

One important thing to point out is that the folks behind Quanteda are *constantly* tinkering with the code, which means that code that works today might not work a few months from now. Even on the Quanteda website they have different code for their Tutorial and for their Reference; and the code is different from when Ken Benoit came here to do a class in October 2016. Which for us means that we will likely need to tinker around a bit to get code to work.

2. Reading in Text Data

2.1. Existing Data Frames

We will use the existing US Presidential Inaugural Speeches text from Quanteda as our working example. We will first use the `system.file()` function to set where the example datasets from the `readtext` package exist. Then we will use `readtext()` function to read in the inaugural speeches data.

```
library(quanteda)
```

```
## quanteda version 0.99.22
```

```
## Using 3 of 4 threads for parallel computing
```

```
##
## Attaching package: 'quanteda'
```

```
## The following object is masked from 'package:utils':
##
##     View
```

```
library(readtext)

data_dir <- system.file("extdata/", package = "readtext")

inaug_data <- readtext(paste0(data_dir, "/csv/inaugCorpus.csv"), text_field="texts")
fix(inaug_data)
```

As we see this version is a very small data set with just five speeches.

2.2 Folders of Text Documents

Often we do not have our text data already in an existing spreadsheet type of set-up. Instead, we have individual text documents (e.g., news stories, party platforms) that we have saved in some type of folder. Again following the Quanteda tutorial let's read in the Universal Declaration of Human Rights data using the `readtext()` function. Note that using * tells R to read-in everything in that folder.

```
udhr_data <- readtext(paste0(data_dir, "/txt/UDHR/*"))
fix(udhr_data)
```

We are just touching on the different combinations of how to text into R. For example, Quanteda now allows you read-in pdf and Word documents. To read-in our own data, we likely need to use the Quanteda references for each particular situation.

3. Corpus Management

3.1. Creating and Setting Corpuses

Once we read-in the data, we need to set it as a corpus using the `corpus()` function. Below we create two corpuses, one with only the 5 speeches (`inaug_corp`) and one with all speeches (`full_inaug_corp`).

```
inaug_corp <- corpus(inaug_data)
ndoc(inaug_corp)
```

```
## [1] 5
```

```
summary(inaug_corp)
```

```
## Corpus consisting of 5 documents:
##
##   Text Types Tokens Sentences            doc_id Year  President FirstName
##  text1   625   1540        23 inaugCorpus.csv.1 1789 Washington    George
##  text2    96    147         4 inaugCorpus.csv.2 1793 Washington    George
##  text3   826   2578        37 inaugCorpus.csv.3 1797      Adams      John
##  text4   717   1927        41 inaugCorpus.csv.4 1801  Jefferson    Thomas
##  text5   804   2381        45 inaugCorpus.csv.5 1805  Jefferson    Thomas
##
## Source:  C:/QTA/Day One/* on x86-64 by bf36y
## Created: Mon May 07 15:01:58 2018
## Notes:
```

```
full_inaug_corp <- corpus(data_corpus_inaugural)
ndoc(full_inaug_corp)
```

```
## [1] 58
```

```
summary(full_inaug_corp)
```

```
## Corpus consisting of 58 documents:
##
##             Text Types Tokens Sentences Year  President       FirstName
##  1789-Washington   625   1538        23 1789 Washington          George
##  1793-Washington    96    147         4 1793 Washington          George
##       1797-Adams   826   2578        37 1797      Adams            John
##   1801-Jefferson   717   1927        41 1801  Jefferson          Thomas
##   1805-Jefferson   804   2381        45 1805  Jefferson          Thomas
##     1809-Madison   535   1263        21 1809    Madison           James
##     1813-Madison   541   1302        33 1813    Madison           James
##      1817-Monroe  1040   3680       121 1817     Monroe           James
##      1821-Monroe  1259   4886       129 1821     Monroe           James
##       1825-Adams  1003   3152        74 1825      Adams     John Quincy
##     1829-Jackson   517   1210        25 1829    Jackson          Andrew
##     1833-Jackson   499   1269        29 1833    Jackson          Andrew
##    1837-VanBuren  1315   4165        95 1837  Van Buren          Martin
##    1841-Harrison  1896   9144       210 1841   Harrison   William Henry
##        1845-Polk  1334   5193       153 1845       Polk      James Knox
##      1849-Taylor   496   1179        22 1849     Taylor         Zachary
##      1853-Pierce  1165   3641       104 1853     Pierce        Franklin
##    1857-Buchanan   945   3086        89 1857   Buchanan           James
##     1861-Lincoln  1075   4006       135 1861    Lincoln         Abraham
##     1865-Lincoln   360    776        26 1865    Lincoln         Abraham
##       1869-Grant   485   1235        40 1869      Grant      Ulysses S.
##       1873-Grant   552   1475        43 1873      Grant      Ulysses S.
##       1877-Hayes   831   2716        59 1877      Hayes   Rutherford B.
##    1881-Garfield  1021   3212       111 1881   Garfield        James A.
##   1885-Cleveland   676   1820        44 1885  Cleveland          Grover
##    1889-Harrison  1352   4722       157 1889   Harrison        Benjamin
##   1893-Cleveland   821   2125        58 1893  Cleveland          Grover
##    1897-McKinley  1232   4361       130 1897   McKinley         William
##    1901-McKinley   854   2437       100 1901   McKinley         William
##   1905-Roosevelt   404   1079        33 1905  Roosevelt        Theodore
##        1909-Taft  1437   5822       159 1909       Taft  William Howard
##      1913-Wilson   658   1882        68 1913     Wilson         Woodrow
##      1917-Wilson   549   1656        59 1917     Wilson         Woodrow
##     1921-Harding  1169   3721       148 1921    Harding       Warren G.
##    1925-Coolidge  1220   4440       196 1925   Coolidge          Calvin
##      1929-Hoover  1090   3865       158 1929     Hoover         Herbert
##   1933-Roosevelt   743   2062        85 1933  Roosevelt     Franklin D.
##   1937-Roosevelt   725   1997        96 1937  Roosevelt     Franklin D.
##   1941-Roosevelt   526   1544        68 1941  Roosevelt     Franklin D.
##   1945-Roosevelt   275    647        26 1945  Roosevelt     Franklin D.
##      1949-Truman   781   2513       116 1949     Truman        Harry S.
##  1953-Eisenhower   900   2757       119 1953 Eisenhower       Dwight D.
##  1957-Eisenhower   621   1931        92 1957 Eisenhower       Dwight D.
##     1961-Kennedy   566   1566        52 1961    Kennedy          John F.
##     1965-Johnson   568   1723        93 1965    Johnson   Lyndon Baines
##       1969-Nixon   743   2437       103 1969      Nixon Richard Milhous
##       1973-Nixon   544   2012        68 1973      Nixon Richard Milhous
##      1977-Carter   527   1376        52 1977     Carter           Jimmy
##      1981-Reagan   902   2790       128 1981     Reagan          Ronald
```

```
##      1985-Reagan   925   2921        123 1985     Reagan          Ronald
##        1989-Bush   795   2681        141 1989       Bush          George
##     1993-Clinton   642   1833         81 1993    Clinton            Bill
##     1997-Clinton   773   2449        111 1997    Clinton            Bill
##        2001-Bush   621   1808         97 2001       Bush       George W.
##        2005-Bush   773   2319        100 2005       Bush       George W.
##       2009-Obama   938   2711        110 2009      Obama          Barack
##       2013-Obama   814   2317         88 2013      Obama          Barack
##       2017-Trump   582   1660         88 2017      Trump       Donald J.
##
## Source:  Gerhard Peters and John T. Woolley. The American Presidency Project.
## Created: Tue Jun 13 14:51:47 2017
## Notes:   http://www.presidency.ucsb.edu/inaugurals.php
```

We will come back to what exactly the corpus summary is telling us below.

3.2. Subsetting Corpuses

Sometimes we want to subset the text we have. To do so, we use the `corpus_subset()` function and use one of the column names (variables) to specify exactly what we want. Below we subset it to only include speeches after 1992 with `Year >= 1992`.

```
recent_corp <- corpus_subset(full_inaug_corp, Year >= 1992)
summary(recent_corp)
```

```
## Corpus consisting of 7 documents:
##
##          Text Types Tokens Sentences Year President FirstName
##  1993-Clinton   642   1833        81 1993   Clinton      Bill
##  1997-Clinton   773   2449       111 1997   Clinton      Bill
##     2001-Bush   621   1808        97 2001      Bush George W.
##     2005-Bush   773   2319       100 2005      Bush George W.
##    2009-Obama   938   2711       110 2009     Obama    Barack
##    2013-Obama   814   2317        88 2013     Obama    Barack
##    2017-Trump   582   1660        88 2017     Trump Donald J.
##
## Source:  Gerhard Peters and John T. Woolley. The American Presidency Project.
## Created: Tue Jun 13 14:51:47 2017
## Notes:   http://www.presidency.ucsb.edu/inaugurals.php
```

3.3. Changing the Unit of Analysis

We can change the unit of analysis (e.g., from document to sentence) using the `corpus_reshape()` function and specifying the unit of analysis.

```
sent_inaug_corp <- corpus_reshape(full_inaug_corp, 'sentences')
ndoc(sent_inaug_corp)
```

```
## [1] 5016
```

```
summary(sent_inaug_corp, 10)
```

```
## Corpus consisting of 5016 documents, showing 10 documents:
##
##                Text Types Tokens Sentences Year  President FirstName
```

```
##   1789-Washington.1    39     49         1 1789 Washington    George
##   1789-Washington.2    63     98         1 1789 Washington    George
##   1789-Washington.3    58     75         1 1789 Washington    George
##   1789-Washington.4    32     35         1 1789 Washington    George
##   1789-Washington.5    68     96         1 1789 Washington    George
##   1789-Washington.6    81    112         1 1789 Washington    George
##   1789-Washington.7    36     41         1 1789 Washington    George
##   1789-Washington.8    23     26         1 1789 Washington    George
##   1789-Washington.9    67     94         1 1789 Washington    George
##  1789-Washington.10    21     22         1 1789 Washington    George
##
## Source:  Gerhard Peters and John T. Woolley. The American Presidency Project.
## Created: Mon May 07 15:01:58 2018
## Notes:   corpus_reshape.corpus(full_inaug_corp, "sentences")
```

```
para_inaug_corp <- corpus_reshape(full_inaug_corp, 'paragraphs')
ndoc(para_inaug_corp)
```

```
## [1] 1513
```

```
summary(para_inaug_corp, 10)
```

```
## Corpus consisting of 1513 documents, showing 10 documents:
##
##               Text Types Tokens Sentences Year  President FirstName
##  1789-Washington.1     8     11         1 1789 Washington    George
##  1789-Washington.2   184    342         5 1789 Washington    George
##  1789-Washington.3   192    328         6 1789 Washington    George
##  1789-Washington.4   214    391         5 1789 Washington    George
##  1789-Washington.5   120    182         2 1789 Washington    George
##  1789-Washington.6   102    164         4 1789 Washington    George
##  1789-Washington.7    88    120         1 1789 Washington    George
##  1793-Washington.1    47     64         2 1793 Washington    George
##  1793-Washington.2    61     83         2 1793 Washington    George
##       1797-Adams.1   114    180         2 1797      Adams      John
##
## Source:  Gerhard Peters and John T. Woolley. The American Presidency Project.
## Created: Mon May 07 15:01:59 2018
## Notes:   corpus_reshape.corpus(full_inaug_corp, "paragraphs")
```

3.4. Tokenizing Corpuses

One of the key activities towards actually analysing our corpuses involves **tokenizing**. We were seeing this already previously when creating our corpuses, but a **token** is, usually, a single word - regardless of the length of the word - or a sentence - regardless of the length of the sentence. We normally first create tokens for each word.

We tokenize our corpus using the `tokens()` function. The `tokens()` function removes white space, but we can also ask it to remove punctuation by including the `remove_punct=TRUE` argument. We will then ask for the first 50 tokens from the 1st document.

```
toks <- tokens(inaug_corp, remove_punct=TRUE)
head(toks[[1]],50)
```

```
##  [1] "Fellow-Citizens" "of"              "the"
##  [4] "Senate"          "and"             "of"
```

```
##  [7] "the"             "House"           "of"
## [10] "Representatives" "Among"           "the"
## [13] "vicissitudes"    "incident"        "to"
## [16] "life"            "no"              "event"
## [19] "could"           "have"            "filled"
## [22] "me"              "with"            "greater"
## [25] "anxieties"       "than"            "that"
## [28] "of"              "which"           "the"
## [31] "notification"    "was"             "transmitted"
## [34] "by"              "your"            "order"
## [37] "and"             "received"        "on"
## [40] "the"             "14th"            "day"
## [43] "of"              "the"             "present"
## [46] "month"           "On"              "the"
## [49] "one"             "hand"
```

3.5. Removing Stopwords

We normally want to get rid of words that do not convey much substantive meaning, such as 'the', 'a', 'of', etc. These are typically referred to use as stopwords. We can remove them using the `tokens_select()` function.

```
toks1 <- tokens_select(toks, stopwords('english'), selection='remove')
head(toks1[[1]],50)
```

```
##  [1] "Fellow-Citizens" "Senate"          "House"
##  [4] "Representatives" "Among"           "vicissitudes"
##  [7] "incident"        "life"            "event"
## [10] "filled"          "greater"         "anxieties"
## [13] "notification"    "transmitted"     "order"
## [16] "received"        "14th"            "day"
## [19] "present"         "month"           "one"
## [22] "hand"            "summoned"        "Country"
## [25] "whose"           "voice"           "can"
## [28] "never"           "hear"            "veneration"
## [31] "love"            "retreat"         "chosen"
## [34] "fondest"         "predilection"    "flattering"
## [37] "hopes"           "immutable"       "decision"
## [40] "asylum"          "declining"       "years"
## [43] "retreat"         "rendered"        "every"
## [46] "day"             "necessary"       "well"
## [49] "dear"            "addition"
```

The default stopwords in Quanteda are very conservative - not extensive - and we often want to exclude additional words that represent little meaning. This is particularly the case when working with topic models. We can do this by adding the word(s) we want to remove using the `c()` function; it seems in the last version of Quanteda that capitalisation of words does not matter. Note: adding stopwords with the `tokens_select()` function (or the more specific `tokens_remove()` function) has always been a bit tempermental though maybe the updates have sorted this out.[1] Below we remove the word 'Among' from our corpus.

```
toks2 <- tokens_select(toks, c(stopwords('english'),'Among'), selection='remove')
head(toks2[[1]],50)
```

```
##  [1] "Fellow-Citizens" "Senate"          "House"
##  [4] "Representatives" "vicissitudes"    "incident"
```

[1] This is an instance where my code is not working with the online tutorial.

```
##  [7] "life"            "event"           "filled"
## [10] "greater"         "anxieties"       "notification"
## [13] "transmitted"     "order"           "received"
## [16] "14th"            "day"             "present"
## [19] "month"           "one"             "hand"
## [22] "summoned"        "Country"         "whose"
## [25] "voice"           "can"             "never"
## [28] "hear"            "veneration"      "love"
## [31] "retreat"         "chosen"          "fondest"
## [34] "predilection"    "flattering"      "hopes"
## [37] "immutable"       "decision"        "asylum"
## [40] "declining"       "years"           "retreat"
## [43] "rendered"        "every"           "day"
## [46] "necessary"       "well"            "dear"
## [49] "addition"        "habit"
```

3.6. Compound Tokens

Often there are phrases (or compound words) that we want to keep as single tokens for our analysis. To do so, we use the `tokens_compound()` function and specify the phrases we want. For example, let's combine House and Representatives.

```
toks3 <- tokens_compound(toks1, phrase(c('House Representatives')))
head(toks3[[1]],50)
```

```
##  [1] "Fellow-Citizens"       "Senate"
##  [3] "House_Representatives" "Among"
##  [5] "vicissitudes"          "incident"
##  [7] "life"                  "event"
##  [9] "filled"                "greater"
## [11] "anxieties"             "notification"
## [13] "transmitted"           "order"
## [15] "received"              "14th"
## [17] "day"                   "present"
## [19] "month"                 "one"
## [21] "hand"                  "summoned"
## [23] "Country"               "whose"
## [25] "voice"                 "can"
## [27] "never"                 "hear"
## [29] "veneration"            "love"
## [31] "retreat"               "chosen"
## [33] "fondest"               "predilection"
## [35] "flattering"            "hopes"
## [37] "immutable"             "decision"
## [39] "asylum"                "declining"
## [41] "years"                 "retreat"
## [43] "rendered"              "every"
## [45] "day"                   "necessary"
## [47] "well"                  "dear"
## [49] "addition"              "habit"
```

3.7. N-Grams

We can create n-grams using the `tokens_ngrams()` function. N-grams are simply sequences of words from already created tokens. For example, bi-grams are two-word phrases, tri-grams are three-word phrases, etc. These are only really useful when you have a specific reason to look at them. We use the option of `n=` to classify the type of n-grams we want. Below we first create bi-grams and second we create tri-grams.

```
ngram <- tokens_ngrams(toks1, n=2)
head(ngram[[1]],50)
```

```
##  [1] "Fellow-Citizens_Senate"   "Senate_House"
##  [3] "House_Representatives"    "Representatives_Among"
##  [5] "Among_vicissitudes"       "vicissitudes_incident"
##  [7] "incident_life"            "life_event"
##  [9] "event_filled"             "filled_greater"
## [11] "greater_anxieties"        "anxieties_notification"
## [13] "notification_transmitted" "transmitted_order"
## [15] "order_received"           "received_14th"
## [17] "14th_day"                 "day_present"
## [19] "present_month"            "month_one"
## [21] "one_hand"                 "hand_summoned"
## [23] "summoned_Country"         "Country_whose"
## [25] "whose_voice"              "voice_can"
## [27] "can_never"                "never_hear"
## [29] "hear_veneration"          "veneration_love"
## [31] "love_retreat"             "retreat_chosen"
## [33] "chosen_fondest"           "fondest_predilection"
## [35] "predilection_flattering"  "flattering_hopes"
## [37] "hopes_immutable"          "immutable_decision"
## [39] "decision_asylum"          "asylum_declining"
## [41] "declining_years"          "years_retreat"
## [43] "retreat_rendered"         "rendered_every"
## [45] "every_day"                "day_necessary"
## [47] "necessary_well"           "well_dear"
## [49] "dear_addition"            "addition_habit"
```

```
ngram <- tokens_ngrams(toks1, n=2:3)
head(ngram[[1]],50)
```

```
##  [1] "Fellow-Citizens_Senate"   "Senate_House"
##  [3] "House_Representatives"    "Representatives_Among"
##  [5] "Among_vicissitudes"       "vicissitudes_incident"
##  [7] "incident_life"            "life_event"
##  [9] "event_filled"             "filled_greater"
## [11] "greater_anxieties"        "anxieties_notification"
## [13] "notification_transmitted" "transmitted_order"
## [15] "order_received"           "received_14th"
## [17] "14th_day"                 "day_present"
## [19] "present_month"            "month_one"
## [21] "one_hand"                 "hand_summoned"
## [23] "summoned_Country"         "Country_whose"
## [25] "whose_voice"              "voice_can"
## [27] "can_never"                "never_hear"
## [29] "hear_veneration"          "veneration_love"
## [31] "love_retreat"             "retreat_chosen"
```

```
## [33] "chosen_fondest"           "fondest_predilection"
## [35] "predilection_flattering"  "flattering_hopes"
## [37] "hopes_immutable"          "immutable_decision"
## [39] "decision_asylum"          "asylum_declining"
## [41] "declining_years"          "years_retreat"
## [43] "retreat_rendered"         "rendered_every"
## [45] "every_day"                "day_necessary"
## [47] "necessary_well"           "well_dear"
## [49] "dear_addition"            "addition_habit"
```

```
tail(ngram[[1]],50)
```

```
##  [1] "sentiments_awakened_occasion"
##  [2] "awakened_occasion_brings"
##  [3] "occasion_brings_us"
##  [4] "brings_us_together"
##  [5] "us_together_shall"
##  [6] "together_shall_take"
##  [7] "shall_take_present"
##  [8] "take_present_leave"
##  [9] "present_leave_without"
## [10] "leave_without_resorting"
## [11] "without_resorting_benign"
## [12] "resorting_benign_Parent"
## [13] "benign_Parent_Human"
## [14] "Parent_Human_Race"
## [15] "Human_Race_humble"
## [16] "Race_humble_supplication"
## [17] "humble_supplication_since"
## [18] "supplication_since_pleased"
## [19] "since_pleased_favor"
## [20] "pleased_favor_American"
## [21] "favor_American_people"
## [22] "American_people_opportunities"
## [23] "people_opportunities_deliberating"
## [24] "opportunities_deliberating_perfect"
## [25] "deliberating_perfect_tranquillity"
## [26] "perfect_tranquillity_dispositions"
## [27] "tranquillity_dispositions_deciding"
## [28] "dispositions_deciding_unparalleled"
## [29] "deciding_unparalleled_unanimity"
## [30] "unparalleled_unanimity_form"
## [31] "unanimity_form_government"
## [32] "form_government_security"
## [33] "government_security_union"
## [34] "security_union_advancement"
## [35] "union_advancement_happiness"
## [36] "advancement_happiness_divine"
## [37] "happiness_divine_blessing"
## [38] "divine_blessing_may"
## [39] "blessing_may_equally"
## [40] "may_equally_conspicuous"
## [41] "equally_conspicuous_enlarged"
## [42] "conspicuous_enlarged_views"
## [43] "enlarged_views_temperate"
```

```
## [44] "views_temperate_consultations"
## [45] "temperate_consultations_wise"
## [46] "consultations_wise_measures"
## [47] "wise_measures_success"
## [48] "measures_success_Government"
## [49] "success_Government_must"
## [50] "Government_must_depend"
```

3.8. Document-Feature Matrix (DFM)

To carry out various statistical analyses with Quanteda, we need to convert our corpuses or tokens to a **Document-Feature Matrix (DFM)**; outside of Quanteda, a DFM is usually called a **Document-Text Matrix (DTM)**. The simplest way to think of DFMs is as *data frames* we traditionally have for statstical analysis in R. To convert our corpus to a DFM we use the `dfm()` function.

```
dfm1 <- dfm(toks1)
head(dfm1,5)
```

```
## Document-feature matrix of: 5 documents, 6 features (56.7% sparse).
## 5 x 6 sparse Matrix of class "dfm"
##        features
## docs    fellow-citizens senate house representatives among vicissitudes
##   text1               1      1     2               2     1            1
##   text2               0      0     0               0     0            0
##   text3               3      1     0               2     4            0
##   text4               2      0     0               0     1            0
##   text5               0      0     0               0     7            0
```

```
dfm1
```

```
## Document-feature matrix of: 5 documents, 1,826 features (72.2% sparse).
```

We see that there are 5 documents and 1,826 features (words) in this dfm.

Since the above only includes the 5 speeches, let's convert the full speech corpus to a dfm. Just to demonstrate, we ask for the first 50 words from the 10th speech in the `head()` function.

```
full_toks <- tokens(data_corpus_inaugural, remove_punct=TRUE)
full_toks <- tokens_select(full_toks, stopwords('english'), selection='remove')
head(full_toks[[10]],50)
```

```
##  [1] "compliance"   "usage"        "coeval"       "existence"
##  [5] "Federal"      "Constitution" "sanctioned"   "example"
##  [9] "predecessors" "career"       "upon"         "enter"
## [13] "appear"       "fellow"       "citizens"     "presence"
## [17] "Heaven"       "bind"         "solemnities"  "religious"
## [21] "obligation"   "faithful"     "performance"  "duties"
## [25] "allotted"     "station"      "called"       "unfolding"
## [29] "countrymen"   "principles"   "shall"        "governed"
## [33] "fulfillment"  "duties"       "first"        "resort"
## [37] "Constitution" "shall"        "swear"        "best"
## [41] "ability"      "preserve"     "protect"      "defend"
## [45] "revered"      "instrument"   "enumerates"   "powers"
## [49] "prescribes"   "duties"
```

```
dfm2 <- dfm(full_toks)
dfm2
```

```
## Document-feature matrix of: 58 documents, 9,205 features (92.6% sparse).
```

We see this new dfm includes 58 documents (speeches) and includes 9,205 features (words).

3.9. Trimming a DFM

Researchers commonly *trim* dfms (or corpuses) to remove words that occur very commonly and/or words that occur very rarely. Trimming is akin to using stopwords, but with a much wider brush. Whether to trim and by how much is very much an art and researcher decision. We may find in our analysis that certain words show up all the time, but don't differentiate the texts in anyway and so we might want to trim them, etc. So, like stopwords, we might come back to trimming after we have carried out our initial analysis.

We also trim a dfm in order to select certain texts within our corpus. This valuable if we only want to look at certain variables, etc., in our corpus. For example, we will do this below by selecting only certain speeches to analyse.

To trim, we use the `dfm_trim()` function and specify how we want to trim. For example, if we want to get of words (features) that only occur 5 times or less, we do the following:[2]

```
freq_dfm2a <- dfm_trim(dfm2, min_count=5)
freq_dfm2a
```

```
## Document-feature matrix of: 58 documents, 2,584 features (80.7% sparse).
```

We now see that the dfm only has 2,584 features (words).

We can do the max version of this and get read of words occuring more than 10 times using `max_count=`.

```
freq_dfm2b <- dfm_trim(dfm2, max_count=10)
freq_dfm2b
```

```
## Document-feature matrix of: 58 documents, 7,960 features (96.1% sparse).
```

We now see that the dfm only has 7,960 features (words).

4. Simple Analysis

Ultimately, the analysis we choose will depend on our corpus and substantive interests. Let's start by looking at some of the more basic types of analysis.

4.1. Top Features in a DFM

We can quickly look at the top features - most common words - in a dfm using the `topfeatures()` function and we ask for the top 25 words.

```
topfeatures(dfm2, 25)
```

```
##       people   government           us          can         upon
##          575          564          478          471          371
##         must        great          may       states        shall
##          366          340          338          333          314
##        world      country        every       nation        peace
##          311          304          298          293          254
##          one          new        power       public          now
```

[2] This is an instance where my code is not working with the online tutorial.

```
##          252          247          236          224          224
##         time     citizens constitution       united      nations
##          216          208          206          202          199
```

To get top features as a proportion, we use the `dfm_weight()` function and include the option `type="relfreq"`.[3]

```
prop_dfm2 <- dfm_weight(dfm2, type = "relfreq")
topfeatures(prop_dfm2, 25)
```

```
##     people         us government        can      shall       must
##  0.5160609  0.4913294  0.4748503  0.4235677  0.3608267  0.3324509
##      world     nation       upon        may      great    country
##  0.3098562  0.3010765  0.2991243  0.2942578  0.2855029  0.2849713
##      every        now     states      peace        new        one
##  0.2692039  0.2516009  0.2484243  0.2464250  0.2410873  0.2084525
##     public    america    nations       time   citizens     united
##  0.2067097  0.2002759  0.1988867  0.1987709  0.1973336  0.1894620
##      power
##  0.1865586
```

This tells us the top words as a proportion of all words (features).

4.2. Simple Frequency Analysis

Like top features, we can examine the most frequent words used. Let's examine `dfm2` using the `textstat_frequency()` function. This function gives us the frequency of words, the rank, and how many documents the word appears in. For plotting purposes, we will let the analysis to the 20 most frequent words.

```
freq <- textstat_frequency(dfm2, n=20)
head(freq,20)
```

```
##       feature frequency rank docfreq
## 1      people       575    1      56
## 2  government       564    2      52
## 3          us       478    3      55
## 4         can       471    4      55
## 5        upon       371    5      47
## 6        must       366    6      51
## 7       great       340    7      55
## 8         may       338    8      53
## 9      states       333    9      46
## 10      shall       314   10      50
## 11      world       311   11      52
## 12    country       304   12      53
## 13      every       298   13      51
## 14     nation       293   14      53
## 15      peace       254   15      46
## 16        one       252   16      48
## 17        new       247   17      49
## 18      power       236   18      47
## 19     public       224   19      41
## 20        now       224   20      52
```

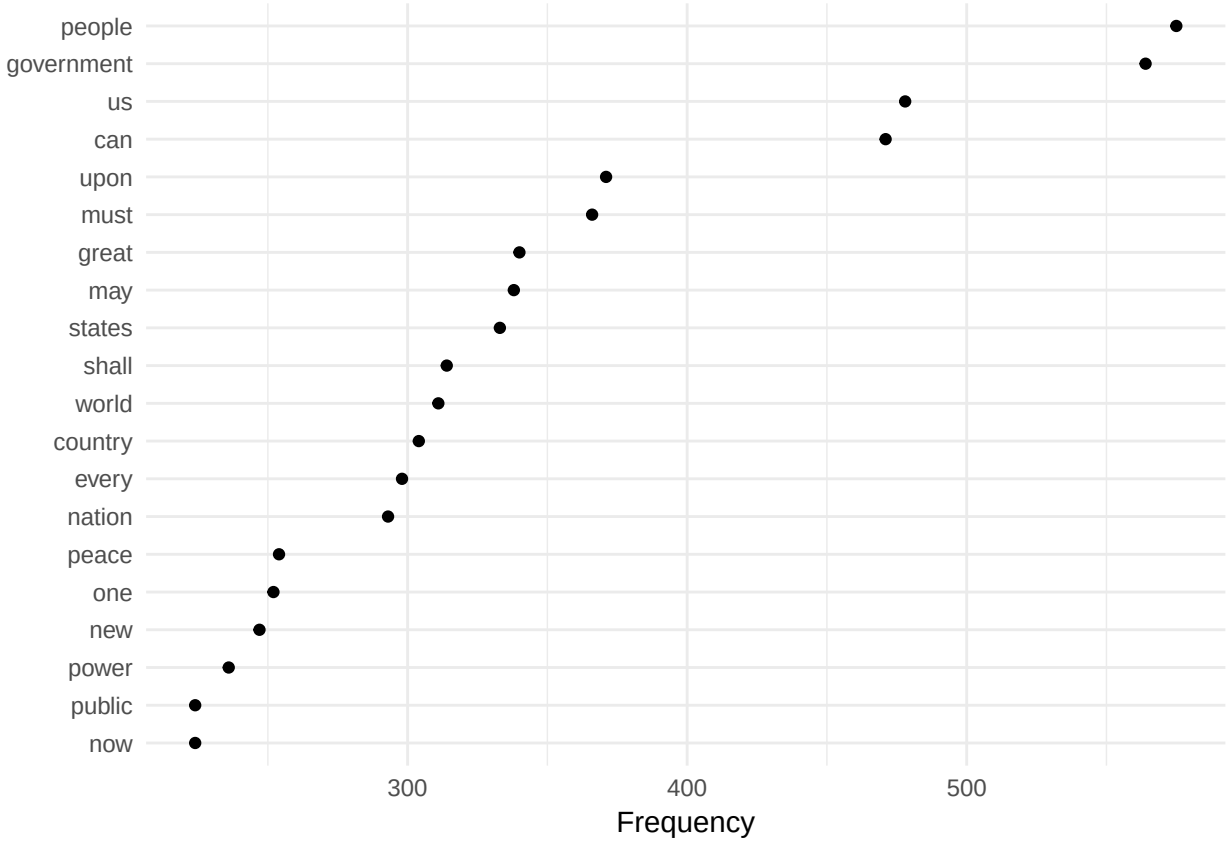
[3]This is an instance where my code is not working with the online tutorial.

For example, we see that 'people' appears 575 times in 56 out of our 58 documents.

We can plot the frequencies using `ggplot2` code.

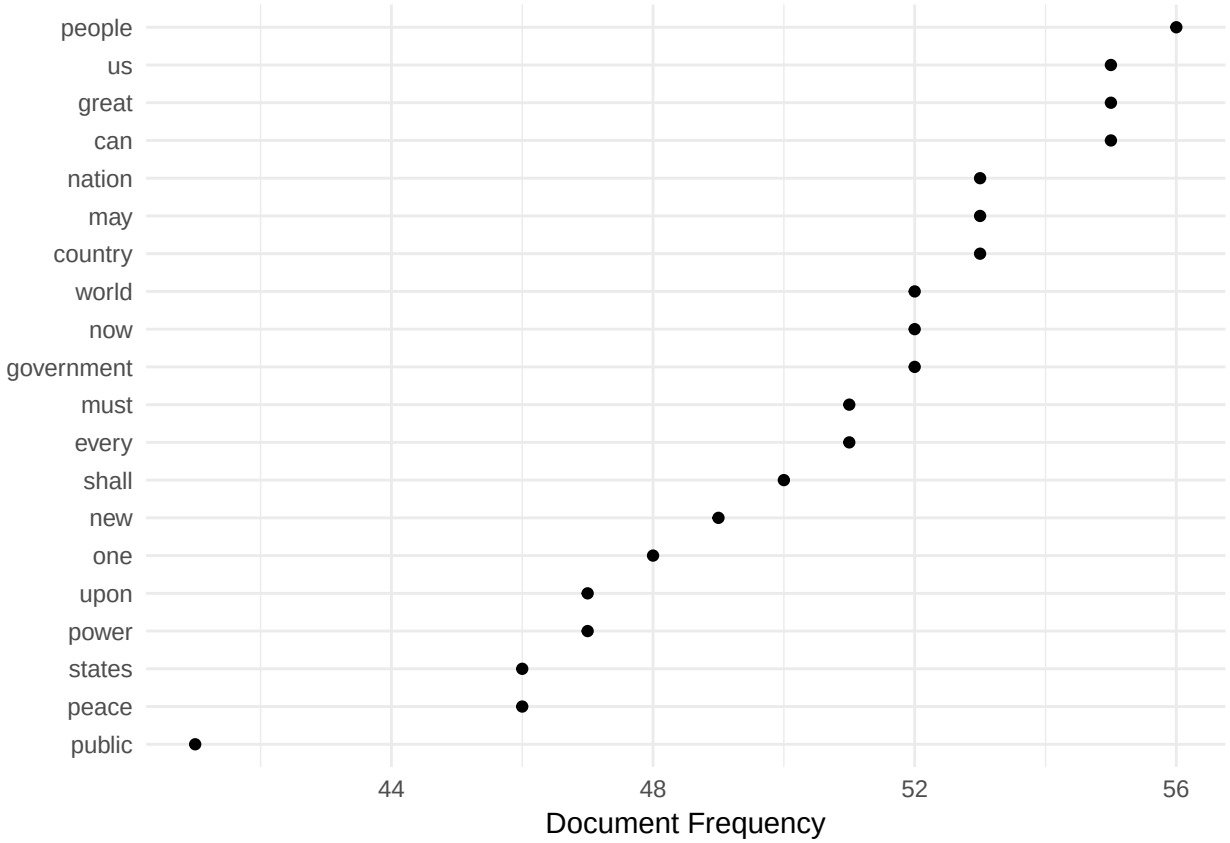
```
library(ggplot2)

x11()
ggplot(data=freq, aes(x = reorder(feature, frequency), y = frequency)) +
  geom_point() +
  coord_flip() +
  labs(x = NULL, y = "Frequency")  +
  theme_minimal()
```



We can also plot the document frequencies using `ggplot2` code.

```
x11()
ggplot(data=freq, aes(x = reorder(feature, docfreq), y = docfreq)) +
  geom_point() +
  coord_flip() +
  labs(x = NULL, y = "Document Frequency") +
  theme_minimal()
```



Below we plot the proportion (relative frequency) for Bush, Obama, and Trump.[4] We could just subset our existing corpus, but instead let's do the version that Quanteda has where we re-read the data and perform all the manipulation in one set of code. This version uses code from `dplyr` where %>% represents continuity in the code (called pipes). This means we are telling R to do all four functions at once and save it as the object `dfm_weight_pres`. We could do this in four separate steps, but this version is considered more elegant.

```
dfm_weight_pres <- data_corpus_inaugural %>%
     corpus_subset(Year > 2000) %>%
     dfm(remove = stopwords("english"), remove_punct = TRUE) %>%
     dfm_weight(type = "relfreq")
```

Now we use the `textstate_frequency` function to calculate the relative frequency by president and we ask for the 10 top words with `n=10`. We can look at all 3 sets of 10 top words using the `head()` function.

```
freq_weight <- textstat_frequency(dfm_weight_pres, n = 10, groups = "President")
head(freq_weight,30)
```

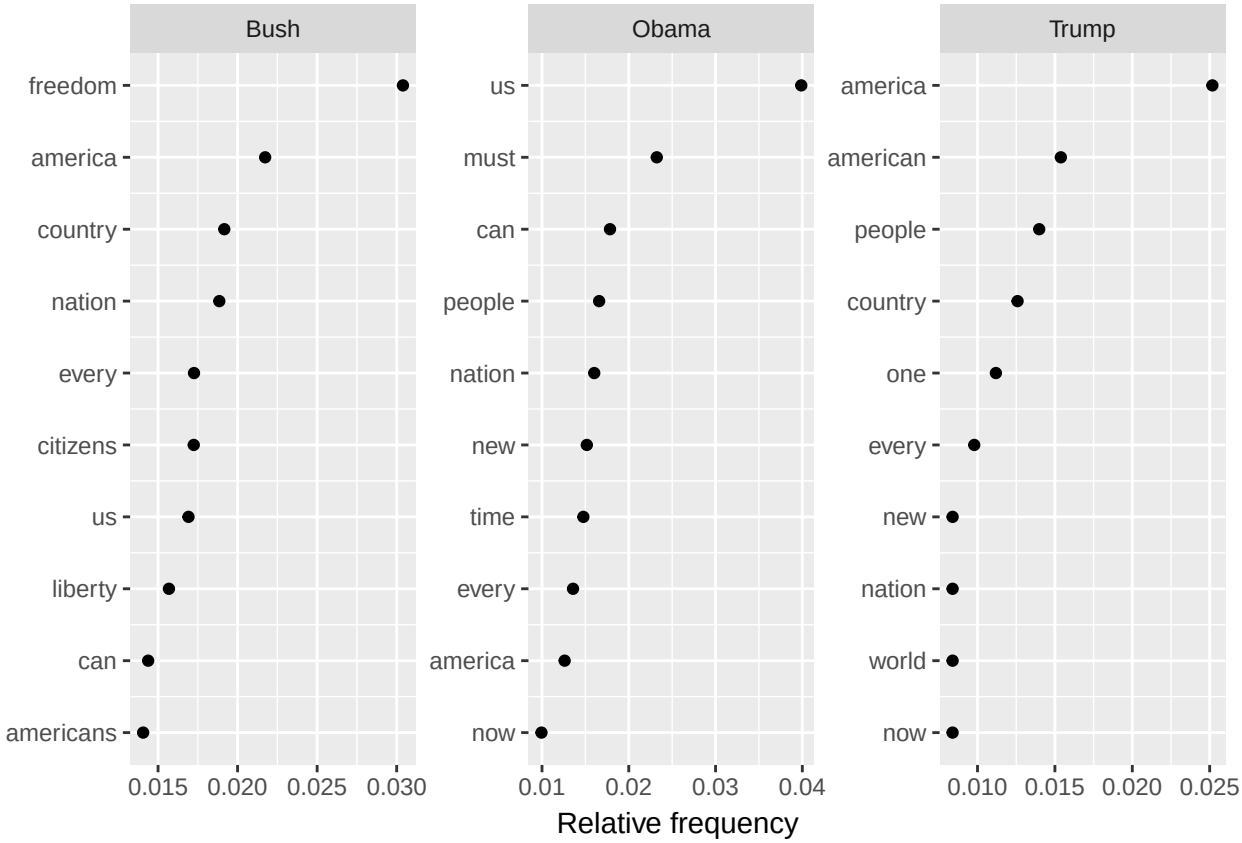
```
##      feature   frequency rank docfreq group
## 1    freedom 0.030392921    1       2  Bush
## 2    america 0.021731459    2       2  Bush
## 3    country 0.019164510    3       2  Bush
## 4     nation 0.018849615    4       2  Bush
## 5      every 0.017259209    5       2  Bush
## 6   citizens 0.017243281    6       2  Bush
## 7         us 0.016912457    7       2  Bush
## 8    liberty 0.015684732    8       2  Bush
```

[4] This is an instance where my code is not working with the online tutorial.

```
## 9        can 0.014377365    9       2  Bush
## 10 americans 0.014062469   10       2  Bush
## 11        us 0.039878113    1       2 Obama
## 12      must 0.023229386    2       2 Obama
## 13       can 0.017839451    3       2 Obama
## 14    people 0.016585365    4       2 Obama
## 15    nation 0.016021685    5       2 Obama
## 16       new 0.015169170    6       2 Obama
## 17      time 0.014767599    7       2 Obama
## 18     every 0.013576876    8       2 Obama
## 19   america 0.012611625    9       2 Obama
## 20       now 0.009941344   10       2 Obama
## 21   america 0.025174825    1       1 Trump
## 22  american 0.015384615    2       1 Trump
## 23    people 0.013986014    3       1 Trump
## 24   country 0.012587413    4       1 Trump
## 25       one 0.011188811    5       1 Trump
## 26     every 0.009790210    6       1 Trump
## 27       new 0.008391608    7       1 Trump
## 28    nation 0.008391608    8       1 Trump
## 29     world 0.008391608    9       1 Trump
## 30       now 0.008391608   10       1 Trump
```

Now we use `ggplot` code to plot.

```
x11()
ggplot(data = freq_weight, aes(x = nrow(freq_weight):1, y = frequency)) +
     geom_point() +
     facet_wrap(~ group, scales = "free") +
     coord_flip() +
     scale_x_continuous(breaks = nrow(freq_weight):1,
                        labels = freq_weight$feature) +
     labs(x = NULL, y = "Relative frequency")
```



4.3. Word Clouds

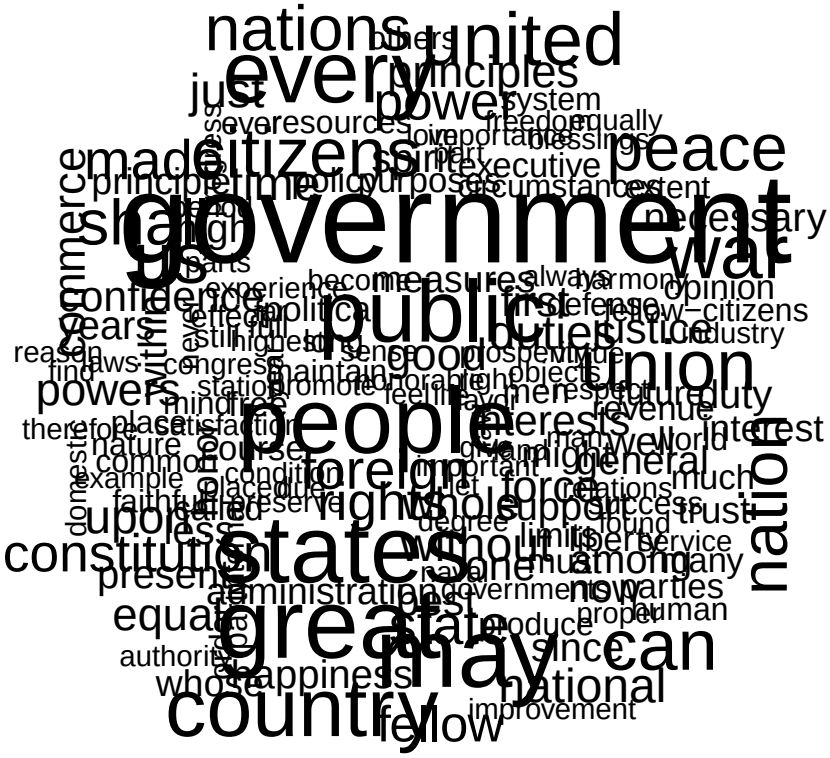
People love word clouds, which one their own are not that useful. Let's just look at the first 10 speeches using the `textplot_wordcloud()` function and specify we only want words that occur at least 10 times. Since the \verb!textplot_wordcloud() is carried out randomly, we will set the seed in order to have exact replications. [5]

```
dfm_inaug <- corpus_subset(data_corpus_inaugural, Year <= 1826) %>%
  dfm(remove = stopwords('english'), remove_punct = TRUE) %>%
  dfm_trim(min_count = 10)
ndoc(dfm_inaug)
```

```
## [1] 10
```

```
set.seed(123)
x11()
textplot_wordcloud(dfm_inaug)
```

[5] This is an instance where my code is not working with the online tutorial.



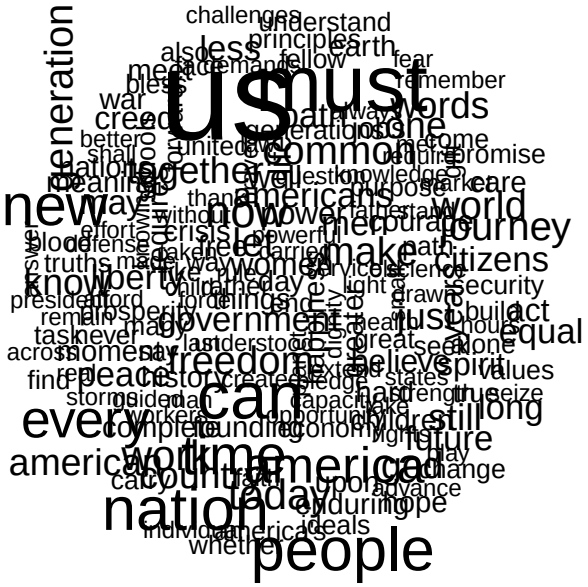
We can add colours by including the `colors=` option.

```
set.seed(123)
x11()
textplot_wordcloud(dfm_inaug,
     color = c('red', 'pink', 'green', 'purple', 'orange', 'blue'))
```

Let's create a word cloud for just Obama by specifying 'Obama' in the `corpus_subset()` function.

```
obama_dfm <-
    dfm(corpus_subset(data_corpus_inaugural, President == "Obama"),
        remove = stopwords("english"), remove_punct = TRUE) %>%
    dfm_trim(min_count = 3)

set.seed(10)
x11()
textplot_wordcloud(obama_dfm)
```



We can also create word clouds that compare several different 'groups'; up to 8 different groups. Besides from creating a subsetted corpus, we need to include the option `comparison=TRUE` in the `textplot_wordcloud` function!. We will also use the `dplyr` code %in%, which specifies the particular presidents we want. Let's compare the first 3 US presidents.

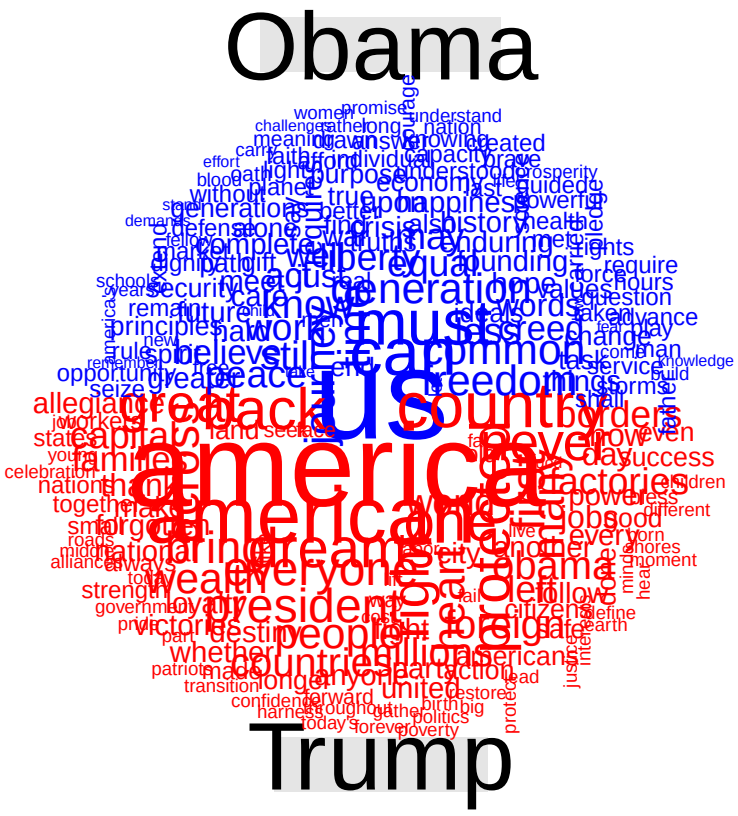
```
first3_dfm <- corpus_subset(data_corpus_inaugural,
               President %in% c("Washington", "Adams", "Jefferson")) %>%
    dfm(groups = "President", remove = stopwords("english"), remove_punct = TRUE) %>%
    dfm_trim(min_count = 10, verbose = FALSE)

set.seed(123)
x11()
textplot_wordcloud(first3_dfm, comparison = TRUE)
```

Finally, let's do a version comparing Obama and Trump.

```
obama_trump_dfm <-
    dfm(corpus_subset(data_corpus_inaugural, President %in% c("Obama", "Trump")),
        remove = stopwords("english"), remove_punct = TRUE, groups = "President") %>%
    dfm_trim(min_count = 3)

set.seed(123)
x11()
textplot_wordcloud(obama_trump_dfm, comparison = TRUE, max_words = 300,
                   color = c("blue", "red"))
```



4.4. Keyword-in-Context

Another simple and common type of analysis is called **Keyword-in-Context** or **KWIC**. This analysis involves specify a keyword or phrase, and identifying the text surrounding the keyword. In its most basic form, KWIC is not very useful since it is difficult to aggregate the results. However, it is good when doing a close read and for exploratory analysis.

We use the `kwic()` function, specify the keyword we are interested in, the number of words (features) to surround the keyword, and how to locate the keywords (using `valuetype=`). First, we will locate `secure*` using `"glob"`, which the combination looks for wild cards of the keyword `secure`.

```
head(kwic(data_corpus_inaugural, "secure*", window = 3, valuetype = "glob"))
```

```
##
##       [1797-Adams, 479]  welfare, and | secure  | the blessings of
##      [1797-Adams, 1513]  nations, and | secured | immortal glory with
##  [1805-Jefferson, 2368]   , and shall | secure  | to you the
##     [1817-Monroe, 1755] cherished. To | secure  | us against these
##     [1817-Monroe, 1815] defense as to | secure  | our cities and
##     [1817-Monroe, 3012]      I can to | secure  | economy and fidelity
```

Next, we want use a regular expression search on the word stem `secur`.

```
head(kwic(data_corpus_inaugural, "secur", window = 3, valuetype = "regex"))
```

```
##
##  [1789-Washington, 1497] government for the | security |
```

```
##        [1797-Adams, 479]       welfare, and |  secure  |
##       [1797-Adams, 1513]       nations, and | secured  |
##   [1805-Jefferson, 2368]        , and shall |  secure  |
##      [1813-Madison, 321]       seas and the | security |
##      [1817-Monroe, 1610]      may form some | security |
##
##  of their union
##  the blessings of
##  immortal glory with
##  to you the
##  of an important
##  against these dangers
```

Compared to our first KWIC, we see that the output shows all words that begin with `secur`.

Finally, if we want an exact keyword, we just include the keyword and set is at `fixed`.

```
head(kwic(data_corpus_inaugural, "secured", window = 3, valuetype = "fixed"))
```

```
##
##     [1797-Adams, 1513]             nations, and | secured |
##    [1821-Monroe, 2083]         of example being | secured |
##    [1821-Monroe, 3671]           soil should be | secured |
##      [1825-Adams, 320]          lot of humanity | secured |
##  [1837-VanBuren, 3227] and prosperity perfectly | secured |
##  [1841-Harrison, 1177]     and personal liberty | secured |
##
##  immortal glory with
##  , policy as
##  to each individual
##  the freedom and
##  . To the
##  to the citizen
```

We can also specify phrases instead of just keywords.

```
head(kwic(data_corpus_inaugural, phrase("personal liberty"), window = 3, valuetype = "fixed"))
```

```
##
##  [1841-Harrison, 1175:1176] preserved, and | personal liberty |
##      [1977-Carter, 564:565] high degree of | personal liberty |
##
##  secured to the
##  , and we
```

5. ON YOUR OWN

Now try replicating the above with your own dataset or the one I have provided.