

## General syntax

- **corpus\_\*** manage text collections/metadata
- **tokens\_\*** create/modify tokenized texts
- **dfm\_\*** create/modify doc-feature matrices
- **fcm\_\*** work with co-occurrence matrices
- **textstat\_\*** calculate text-based statistics
- **textmodel\_\*** fit (un-)supervised models
- **textplot\_\*** create text-based visualizations

### Consistent grammar:

- **object()** constructor for the object type
- **object\_verb()** inputs & returns object type

## Extensions

**quanteda** works well with these companion packages:

- **readtext:** An easy way to read text data
- **spacyr:** NLP using the spaCy library
- **quanteda.data:** additional textual data
- **stopwords:** multilingual stopword lists in R

## Create a corpus from texts (corpus\_\*)

### Read texts (txt, pdf, csv, doc, docx, json, xml)

```
my_texts <- readtext::readtext("~/link/to/path/*")
```

### Construct a corpus from a character vector

```
x <- corpus(data_char_ukimmig2010, text_field = "text")
```

### Explore a corpus

```
summary(data_corpus_inaugural, n = 2)
# Corpus consisting of 58 documents, showing 2 documents:
#      Text Types Tokens Sentences Year  President FirstName
# 1789-Washington 625 1538      23 1789 Washington  George
# 1793-Washington  96  147       4 1793 Washington  George
#
# 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
```

### Extract or add document-level variables

```
party <- docvars(data_corpus_inaugural, "Party")
docvars(x, "serial_number") <- 1:ndoc(x)
```

### Bind or subset corpora

```
corpus(x[1:5]) + corpus(x[7:9])
corpus_subset(x, Year > 1990)
```

### Change units of a corpus

```
corpus_reshape(x, to = c("sentences", "paragraphs"))
```

### Segment texts on a pattern match

```
corpus_segment(x, pattern, valuetype, extract_pattern = TRUE)
```

### Take a random sample of corpus texts

```
corpus_sample(x, size = 10, replace = FALSE)
```

## Extract features (dfm\_\*; fcm\_\*)

### Create a document-feature matrix (dfm) from a corpus

```
x <- dfm(data_corpus_inaugural,
        tolower = TRUE, stem = FALSE, remove_punct = TRUE,
        remove = stopwords("english"))
```

```
head(x, n = 2, nf = 4)
```

```
## Document-feature matrix of: 2 documents, 4 features (41.7% sparse).
##              features
## docs      fellow-citizens senate house representatives
## 1789-Washington      1      1      2      2
## 1793-Washington      0      0      0      0
```

### Create a dictionary

```
dictionary(list(negative = c("bad", "awful", "sad"),
                positive = c("good", "wonderful", "happy")))
```

### Apply a dictionary

```
dfm_lookup(x, dictionary = data_dictionary_LSD2015)
```

### Select features

```
dfm_select(x, dictionary = data_dictionary_LSD2015)
```

### Compress a dfm by combining identical elements

```
dfm_compress(x, margin = c("both", "documents", "features"))
```

### Randomly sample documents or features

```
dfm_sample(x, what = c("documents", "features"))
```

### Weight or smooth the feature frequencies

```
dfm_weight(x, type = "prop") | dfm_smooth(x, smoothing = 0.5)
```

### Sort or group a dfm

```
dfm_sort(x, margin = c("features", "documents", "both"))
dfm_group(x, groups = "President")
```

### Combine identical dimension elements of a dfm

```
dfm_compress(x, margin = c("both", "documents", "features"))
```

### Create a feature co-occurrence matrix (fcm)

```
x <- fcm(data_corpus_inaugural, context = "window", size = 5)
fcm_compress/remove/select/toupper/tolower are also available
```

## Useful additional functions

### Locate keywords-in-context

```
kwic(data_corpus_inaugural, "america*")
```

### Utility functions

<code>texts(corpus)</code>	Show texts of a corpus
<code>ndoc(corpus/dfm/tokens)</code>	Count documents/features
<code>nfeat(corpus/dfm/tokens)</code>	Count features
<code>summary(corpus/dfm)</code>	Print summary
<code>head(corpus/dfm)</code>	Return first part
<code>tail(corpus/dfm)</code>	Return last part

## Tokenize a set of texts (tokens\_\*)

### Tokenize texts from a character vector or corpus

```
x <- tokens("Powerful tool for text analysis.",  
           remove_punct = TRUE, stem = TRUE)
```

### Convert sequences into compound tokens

```
myseqs <- phrase(c("powerful", "tool", "text analysis"))  
tokens_compound(x, myseqs)
```

### Select tokens

```
tokens_select(x, c("powerful", "text"), selection = "keep")
```

### Create ngrams and skipgrams from tokens

```
tokens_ngrams(x, n = 1:3)  
tokens_skipgrams(toks, n = 2, skip = 0:1)
```

### Convert case of tokens

```
tokens_tolower(x) | tokens_topupper(x)
```

### Stem the terms in an object

```
tokens_wordstem(x)
```

## Calculate text statistics (textstat\_\*)

### Tabulate feature frequencies from a dfm

```
textstat_frequency(x) | topfeatures(x)
```

### Identify and score collocations from a tokenized text

```
toks <- tokens(c("quanteda is a pkg for quant text analysis",  
               "quant text analysis is a growing field"))  
textstat_collocations(toks, size = 3, min_count = 2)
```

### Calculate readability of a corpus

```
textstat_readability(data_corpus_inaugural, measure = "Flesch")
```

### Calculate lexical diversity of a dfm

```
textstat_lexdiv(x, measure = "TTR")
```

### Measure distance or similarity from a dfm

```
textstat_simil(x, "2017-Trump", method = "cosine")  
textstat_dist(x, "2017-Trump", margin = "features")
```

### Calculate keyness statistics

```
textstat_keyness(x, target = "2017-Trump")
```

## Fit text models based on a dfm (textmodel\_\*)

### Correspondence Analysis (CA)

```
textmodel_ca(x, threads = 2, sparse = TRUE, residual_floor = 0.1)
```

### Naïve Bayes classifier for texts

```
textmodel_nb(x, y = training_labels, distribution = "multinomial")
```

### Wordscores text model

```
refscores <- c(seq(-1.5, 1.5, .75), NA))  
textmodel_wordscores(data_dfm_lbgexample, refscores)
```

### Wordfish Poisson scaling model

```
textmodel_wordfish(dfm(data_corpus_irishbudget2010), dir = c(6,5))
```

**Textmodel methods:** `predict()`, `coef()`, `summary()`, `print()`

## Plot features or models (textplot\_\*)

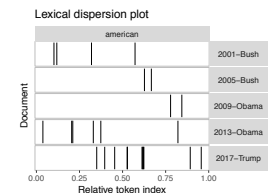
### Plot features as a wordcloud

```
data_corpus_inaugural %>%  
  corpus_subset(President == "Obama") %>%  
  dfm(remove = stopwords("english")) %>%  
  textplot_wordcloud()
```



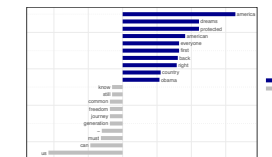
### Plot the dispersion of key word(s)

```
data_corpus_inaugural %>%  
  corpus_subset(Year > 1945) %>%  
  kwic("american") %>%  
  textplot_xray()
```



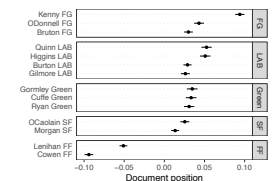
### Plot word keyness

```
data_corpus_inaugural %>%  
  corpus_subset(President %in%  
                c("Obama", "Trump")) %>%  
  dfm(groups = "President",  
       remove = stopwords("english")) %>%  
  textstat_keyness(target = "Trump") %>%  
  textplot_keyness()
```



### Plot Wordfish, Wordscores or CA models

```
textplot_scale1d(scaling_model,  
                 groups = party,  
                 margin = "documents")
```



## Convert dfm to a non-quanteda format

```
convert(x, to = c("lda", "tm", "stm", "austin", "topicmodels",  
                 "lsa", "matrix", "data.frame"))
```