Consistent grammar:

- `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

**General syntax**

**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")`

- Bind or subset corpora
  
  `corpus(x[1:5]) + corpus(x[7:9])`

- 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)`

**Extensions**

- `quanteda` works well with these companion packages:
  
  - `readtext`: an easy way to read text data
  - `spacy`: NLP using the spaCy library
  - `spacyr`: NLP using the spaCy library
  - `corpora`: additional text corpora
  - `stopwords`: multilingual stopword lists in R

**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"))`

- 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)`

- 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"))`

- 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)`

**Useful additional functions**

- Locate keywords-in-context
  
  `kwic(data_corpus_inaugural, "america**")`

**Utility functions**

- `texts(corpus)`
  
  Show texts of a corpus

- `ndoc(corpus/dfm/tokens)`
  
  Count documents/features

- `nfeat(corpus/dfm/tokens)`
  
  Count features

- `summary(corpus/dfm)`
  
  Print summary

- `head(corpus/dfm)`
  
  Return first part

- `tail(corpus/dfm)`
  
  Return last part

[https://creativecommons.org/licenses/by/4.0/]
Fit text models based on a dfm (textmodel_*)

Correspondence Analysis (CA)
textmodel_ca(x, threads = 2, sparse = TRUE, residual_floor = 0.1)

Naive 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(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(Year > 1945) %>%
dfm(threads = 2, sparse = TRUE) %>%
textplot_xray()

Plot the dispersion of key word(s)
data_corpus_inaugural %>%
corpus_subset(Trump) %>%
textplot_xray()

Plot word keyness
data_corpus_inaugural %>%
corpus_subset(Trump) %>%
dfm(groups = "Trump", remove = stopwords("english")) %>%
textstat_keyness(target = "Trump") %>%
textplot_keyness()

Plot 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", "stem", "austins", "topicmodels", "lsa", "matrix", "data.frame"))