quanteda Cheat Sheet

Quantitative Analysis of Textual Data

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:

- **quanteda.textmodels**: Text scaling and classification models
- **readtext**: an easy way to read text data
- **spacyr**: NLP using the spaCy library
- **quanteda.corpora**: additional text corpora
- **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 Party
##  1789-Washington   625   1537        23 1789 Washington    George  none
##  1793-Washington    96    147         4 1793 Washington    George  none
```

Extract or add document-level variables

```
party <- data_corpus_inaugural$Party
x$serial_number <- seq_len(ndoc(x))
docvars(x, "serial_number") <- seq_len(ndoc(x)) # alternative
```

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 = "sentences")
```

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("en"))
```

```
print(x, max_ndoc = 2, max_nfeat = 4)
## Document-feature matrix of: 58 documents, 9,210 features (92.6% sparse) and 4 docvars.
##                  features
## docs              fellow-citizens senate house representatives
##   1789-Washington               1      1     2               2
##   1793-Washington               0      0     0               0
## [ reached max_ndoc ... 56 more documents, reached max_nfeat ... 9,206 more features ]
```

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, pattern = data_dictionary_LSD2015, selection = "keep")
```

Randomly sample documents or features

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

Weight or smooth the feature frequencies

```
dfm_weight(x, scheme = "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, pattern = "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*

Tokenize a set of texts (tokens_*)

Tokenize texts from a character vector or corpus

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

Convert sequences into compound tokens

```
myseqs <- phrase(c("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(x, n = 2, skip = 0:1)
```

Convert case of tokens or features

```
tokens_tolower(x) tokens_toupper(x) dfm_tolower(x)
```

Stem tokens or features

```
tokens_wordstem(x) dfm_wordstem(x)
```

Fit text models based on a dfm (textmodel_*)

These functions require the ***quanteda.textmodels*** *package*

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")
```

SVM classifier for texts

```
textmodel_svm(x, y = training_labels)
```

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()`

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(x, measure = c("Flesch", "FOG"))
```

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",
               margin = c("documents", "features"))
textstat_dist(x, "2017-Trump",
              margin = c("documents", "features"))
```

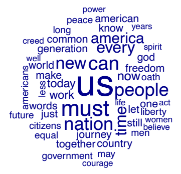
Calculate keyness statistics

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

Plot features or models (textplot_*)

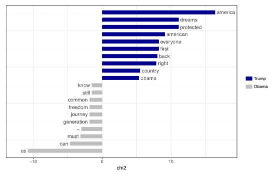
Plot features as a wordcloud

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



Plot word keyness

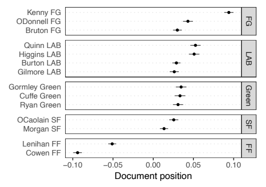
```
data_corpus_inaugural %>%
  corpus_subset(President %in%
                c("Obama", "Trump")) %>%
  dfm(groups = "President",
      remove = stopwords("en")) %>%
  textstat_keyness(target = "Trump") %>%
  textplot_keyness()
```



Plot Wordfish, Wordscores or CA models

(requires the **quanteda.textmodels** package)

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



by **Stefan Müller** and **Kenneth Benoit** • *smueller@quanteda.org, kbenoit@quanteda.org*

Learn more at: http://quanteda.io • updated: 05/2020

Convert dfm to a non-quanteda format

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