Package: quanteda.textstats (via r-universe)

October 28, 2026

Version 0.97.2

Title Textual Statistics for the Quantitative Analysis of Textual Data

Description Textual statistics functions formerly in the 'quanteda' package. Textual statistics for characterizing and comparing textual data. Includes functions for measuring term and document frequency, the co-occurrence of words, similarity and distance between features and documents, feature entropy, keyword occurrence, readability, and lexical diversity. These functions extend the 'quanteda' package and are specially designed for sparse textual data.

License GPL-3

Depends R (>= 3.5.0)

Imports quanteda (>= 4.0.0), Matrix (>= 1.5-0), methods, nsyllable, proxyC (>= 0.1.4), Rcpp (>= 0.12.12), stringi

LinkingTo Rcpp, RcppArmadillo (>= 0.7.600.1.0), quanteda

Suggests entropy, ExPosition, proxy, rmarkdown, spelling, svs, testthat, knitr, covr

URL https://quanteda.io

Encoding UTF-8

BugReports https://github.com/quanteda/quanteda.textstats/issues

LazyData TRUE

Language en-GB

RoxygenNote 7.3.2

NeedsCompilation yes

Author Kenneth Benoit [cre, aut, cph] (<https://orcid.org/0000-0002-0797-564X>), Kohei Watanabe [aut] (<https://orcid.org/0000-0001-6519-5265>), Haiyan Wang [aut] (<https://orcid.org/0000-0003-4992-4311>), Jiong Wei Lua [aut], Jouni Kuha [aut] (<https://orcid.org/0000-0002-1156-8465>), European Research Council [fnd] (ERC-2011-StG 283794-QUANTESS)

Maintainer Kenneth Benoit <kbenoit@lse.ac.uk>

Repository https://cran.r-universe.dev

Date/Publication 2024-09-04 02:53:30 UTC

RemoteUrl https://github.com/cran/quanteda.textstats

RemoteRef HEAD

RemoteSha eea0a541cf5bd14ca03bb56f866c58002b5608ae

Contents

data_char_wordlists 2
textstat_collocations 3
textstat_entropy 5
textstat_frequency 6
textstat_keyness 8
textstat_lexdiv 10
textstat_readability 13
textstat_simil 22
textstat_summary 24

Index **26**

`data_char_wordlists` *Word lists for readability statistics*

Description

`data_char_wordlists` provides word lists used in some readability indexes; it is a named list of character vectors where each list element corresponds to a different readability index.

Usage

```
data_char_wordlists
```

Format

A list of length two:

`DaleChall` The long Dale-Chall list of 3,000 familiar (English) words needed to compute the Dale-Chall Readability Formula.

`Spache` The revised Spache word list (see Klare 1975, 73; Spache 1974) needed to compute the Spache Revised Formula of readability (Spache 1953).

References

Chall, J.S., & Dale, E. (1995). *Readability Revisited: The New Dale-Chall Readability Formula*. Brookline Books.

Dale, E. & Chall, J.S. (1948). A Formula for Predicting Readability. *Educational Research Bulletin*, 27(1): 11–20.

Dale, E. & Chall, J.S. (1948). A Formula for Predicting Readability: Instructions. *Educational Research Bulletin*, 27(2): 37–54.

Klare, G.R. (1975). Assessing Readability. *Reading Research Quarterly* 10(1), 62–102.

Spache, G. (1953). A New Readability Formula for Primary-Grade Reading Materials. *The Elementary School Journal*, 53, 410–413.

Spache, G. (1974). *Good reading for poor readers*. (Rvd. 9th Ed.) Champaign, Illinois: Garrard, 1974.

`textstat_collocations` *Identify and score multi-word expressions*

Description

Identify and score multi-word expressions, or adjacent fixed-length collocations, from text.

Usage

```
textstat_collocations(
  x,
  method = "lambda",
  size = 2,
  min_count = 2,
  smoothing = 0.5,
  tolower = TRUE,
  ...
)
```

Arguments

`x`	a character, corpus, or tokens object whose collocations will be scored. The tokens object should include punctuation, and if any words have been removed, these should have been removed with `padding = TRUE`. While identifying collocations for tokens objects is supported, you will get better results with character or corpus objects due to relatively imperfect detection of sentence boundaries from texts already tokenized.
`method`	association measure for detecting collocations. Currently this is limited to `"lambda"`. See Details.
`size`	integer; the length of the collocations to be scored
`min_count`	numeric; minimum frequency of collocations that will be scored

smoothing	numeric; a smoothing parameter added to the observed counts (default is 0.5)
tolower	logical; if TRUE, form collocations as lower-cased combinations
...	additional arguments passed to tokens()

Details

Documents are grouped for the purposes of scoring, but collocations will not span sentences. If x is a tokens object and some tokens have been removed, this should be done using [tokens_remove](x, pattern, padding = TR so that counts will still be accurate, but the pads will prevent those collocations from being scored.

The lambda computed for a size = K-word target multi-word expression the coefficient for the K-way interaction parameter in the saturated log-linear model fitted to the counts of the terms forming the set of eligible multi-word expressions. This is the same as the "lambda" computed in Blaheta and Johnson's (2001), where all multi-word expressions are considered (rather than just verbs, as in that paper). The z is the Wald z-statistic computed as the quotient of lambda and the Wald statistic for lambda as described below.

In detail:

Consider a K-word target expression x, and let z be any K-word expression. Define a comparison function $c(x, z) = (j_1, \ldots, j_K) = c$ such that the kth element of c is 1 if the kth word in z is equal to the kth word in x, and 0 otherwise. Let $c_i = (j_{i1}, \ldots, j_{iK})$, $i = 1, \ldots, 2^K = M$, be the possible values of $c(x, z)$, with $c_M = (1, 1, \ldots, 1)$. Consider the set of $c(x, z_r)$ across all expressions z_r in a corpus of text, and let n_i, for $i = 1, \ldots, M$, denote the number of the $c(x, z_r)$ which equal c_i, plus the smoothing constant smoothing. The n_i are the counts in a 2^K contingency table whose dimensions are defined by the c_i.

λ: The K-way interaction parameter in the saturated loglinear model fitted to the n_i. It can be calculated as

$$\lambda = \sum_{i=1}^{M} (-1)^{K-b_i} * log n_i$$

where b_i is the number of the elements of c_i which are equal to 1.

Wald test z-statistic z is calculated as:

$$z = \frac{\lambda}{[\sum_{i=1}^{M} n_i^{-1}]^{(1/2)}}$$

Value

textstat_collocations returns a data.frame of collocations and their scores and statistics. This consists of the collocations, their counts, length, and λ and z statistics. When size is a vector, then count_nested counts the lower-order collocations that occur within a higher-order collocation (but this does not affect the statistics).

Author(s)

Kenneth Benoit, Jouni Kuha, Haiyan Wang, and Kohei Watanabe

References

Blaheta, D. & Johnson, M. (2001). Unsupervised learning of multi-word verbs. Presented at the ACLEACL Workshop on the Computational Extraction, Analysis and Exploitation of Collocations.

Examples

```
library("quanteda")
corp <- data_corpus_inaugural[1:2]
head(cols <- textstat_collocations(corp, size = 2, min_count = 2), 10)
head(cols <- textstat_collocations(corp, size = 3, min_count = 2), 10)

# extracting multi-part proper nouns (capitalized terms)
toks1 <- tokens(data_corpus_inaugural)
toks2 <- tokens_remove(toks1, pattern = stopwords("english"), padding = TRUE)
toks3 <- tokens_select(toks2, pattern = "^([A-Z][a-z\\-]{2,})", valuetype = "regex",
                       case_insensitive = FALSE, padding = TRUE)
tstat <- textstat_collocations(toks3, size = 3, tolower = FALSE)
head(tstat, 10)

# vectorized size
txt <- c(". . . . a b c . . a b c . . . c d e",
         "a b . . a b . . a b . . a b . a b",
         "b c d . . b c . b c . . . b c")
textstat_collocations(txt, size = 2:3)

# compounding tokens from collocations
toks <- tokens("This is the European Union.")
colls <- tokens("The new European Union is not the old European Union.") %>%
    textstat_collocations(size = 2, min_count = 1, tolower = FALSE)
colls
tokens_compound(toks, colls, case_insensitive = FALSE)

#' # from a collocations object
(coll <- textstat_collocations(tokens("a b c a b d e b d a b")))
phrase(coll)
```

`textstat_entropy` *Compute entropies of documents or features*

Description

Compute entropies of documents or features

Usage

```
textstat_entropy(x, margin = c("documents", "features"), base = 2)
```

Arguments

x	a dfm
margin	character indicating for which margin to compute entropy
base	base for logarithm function

Value

a data.frame of entropies for the given document or feature

Examples

```
library("quanteda")
textstat_entropy(data_dfm_lbgexample)
textstat_entropy(data_dfm_lbgexample, "features")
```

---

textstat_frequency *Tabulate feature frequencies*

Description

Produces counts and document frequencies summaries of the features in a dfm, optionally grouped by a docvars variable or other supplied grouping variable.

Usage

```
textstat_frequency(
  x,
  n = NULL,
  groups = NULL,
  ties_method = c("min", "average", "first", "random", "max", "dense"),
  ...
)
```

Arguments

x	a dfm object
n	(optional) integer specifying the top n features to be returned, within group if groups is specified
groups	grouping variable for sampling, equal in length to the number of documents. This will be evaluated in the docvars data.frame, so that docvars may be referred to by name without quoting. This also changes previous behaviours for groups. See news(Version >= "3.0", package = "quanteda") for details.
ties_method	character string specifying how ties are treated. See base::rank() for details. Unlike that function, however, the default is "min", so that frequencies of 10, 10, 11 would be ranked 1, 1, 3.
...	additional arguments passed to dfm_group(). This can be useful in passing force = TRUE, for instance, if you are grouping a dfm that has been weighted.

Value

a data.frame containing the following variables:

`feature` (character) the feature

`frequency` count of the feature

`rank` rank of the feature, where 1 indicates the greatest frequency

`docfreq` document frequency of the feature, as a count (the number of documents in which this feature occurred at least once)

`docfreq` document frequency of the feature, as a count

`group` (only if `groups` is specified) the label of the group. If the features have been grouped, then all counts, ranks, and document frequencies are within group. If groups is not specified, the `group` column is omitted from the returned data.frame.

`textstat_frequency` returns a data.frame of features and their term and document frequencies within groups.

Examples

```
library("quanteda")
set.seed(20)
dfmat1 <- dfm(tokens(c("a a b b c d", "a d d d", "a a a")))

textstat_frequency(dfmat1)
textstat_frequency(dfmat1, groups = c("one", "two", "one"), ties_method = "first")
textstat_frequency(dfmat1, groups = c("one", "two", "one"), ties_method = "average")

dfmat2 <- corpus_subset(data_corpus_inaugural, President == "Obama") %>%
   tokens(remove_punct = TRUE) %>%
   tokens_remove(stopwords("en")) %>%
   dfm()
tstat1 <- textstat_frequency(dfmat2)
head(tstat1, 10)

dfmat3 <- head(data_corpus_inaugural) %>%
   tokens(remove_punct = TRUE) %>%
   tokens_remove(stopwords("en")) %>%
   dfm()
textstat_frequency(dfmat3, n = 2, groups = President)


## Not run:
# plot 20 most frequent words
library("ggplot2")
ggplot(tstat1[1:20, ], aes(x = reorder(feature, frequency), y = frequency)) +
    geom_point() +
    coord_flip() +
    labs(x = NULL, y = "Frequency")

# plot relative frequencies by group
dfmat3 <- data_corpus_inaugural %>%
```

```
    corpus_subset(Year > 2000) %>%
    tokens(remove_punct = TRUE) %>%
    tokens_remove(stopwords("en")) %>%
    dfm() %>%
    dfm_group(groups = President) %>%
    dfm_weight(scheme = "prop")

# calculate relative frequency by president
tstat2 <- textstat_frequency(dfmat3, n = 10, groups = President)

# plot frequencies
ggplot(data = tstat2, aes(x = factor(nrow(tstat2):1), y = frequency)) +
    geom_point() +
    facet_wrap(~ group, scales = "free") +
    coord_flip() +
    scale_x_discrete(breaks = nrow(tstat2):1,
                     labels = tstat2$feature) +
    labs(x = NULL, y = "Relative frequency")

## End(Not run)
```

`textstat_keyness` *Calculate keyness statistics*

Description

Calculate "keyness", a score for features that occur differentially across different categories. Here, the categories are defined by reference to a "target" document index in the dfm, with the reference group consisting of all other documents.

Usage

```
textstat_keyness(
  x,
  target = 1L,
  measure = c("chi2", "exact", "lr", "pmi"),
  sort = TRUE,
  correction = c("default", "yates", "williams", "none"),
  ...
)
```

Arguments

`x`	a dfm containing the features to be examined for keyness
`target`	the document index (numeric, character or logical) identifying the document forming the "target" for computing keyness; all other documents' feature frequencies will be combined for use as a reference

measure	(signed) association measure to be used for computing keyness. Currently available: "chi2"; "exact" (Fisher's exact test); "lr" for the likelihood ratio; "pmi" for pointwise mutual information. Note that the "exact" test is very computationally intensive and therefore much slower than the other methods.
sort	logical; if TRUE sort features scored in descending order of the measure, otherwise leave in original feature order
correction	if "default", Yates correction is applied to "chi2"; William's correction is applied to "lr"; and no correction is applied for the "exact" and "pmi" measures. Specifying a value other than the default can be used to override the defaults, for instance to apply the Williams correction to the chi2 measure. Specifying a correction for the "exact" and "pmi" measures has no effect and produces a warning.
...	not used

Value

a data.frame of computed statistics and associated p-values, where the features scored name each row, and the number of occurrences for both the target and reference groups. For measure = "chi2" this is the chi-squared value, signed positively if the observed value in the target exceeds its expected value; for measure = "exact" this is the estimate of the odds ratio; for measure = "lr" this is the likelihood ratio $G2$ statistic; for "pmi" this is the pointwise mutual information statistics.

textstat_keyness returns a data.frame of features and their keyness scores and frequency counts.

References

Bondi, M. & Scott, M. (eds) (2010). *Keyness in Texts*. Amsterdam, Philadelphia: John Benjamins.

Stubbs, M. (2010). Three Concepts of Keywords. In *Keyness in Texts*, Bondi, M. & Scott, M. (eds): 1–42. Amsterdam, Philadelphia: John Benjamins.

Scott, M. & Tribble, C. (2006). *Textual Patterns: Keyword and Corpus Analysis in Language Education*. Amsterdam: Benjamins: 55.

Dunning, T. (1993). Accurate Methods for the Statistics of Surprise and Coincidence. *Computational Linguistics*, 19(1): 61–74.

Examples

```
library("quanteda")

# compare pre- v. post-war terms using grouping
period <- ifelse(docvars(data_corpus_inaugural, "Year") < 1945, "pre-war", "post-war")
dfmat1 <- tokens(data_corpus_inaugural) %>%
    dfm() %>%
    dfm_group(groups = period)
head(dfmat1) # make sure 'post-war' is in the first row
head(tstat1 <- textstat_keyness(dfmat1), 10)
tail(tstat1, 10)

# compare pre- v. post-war terms using logical vector
dfmat2 <- dfm(tokens(data_corpus_inaugural))
```

```
head(textstat_keyness(dfmat2, docvars(data_corpus_inaugural, "Year") >= 1945), 10)

# compare Trump 2017 to other post-war preseidents
dfmat3 <- dfm(tokens(corpus_subset(data_corpus_inaugural, period == "post-war")))
head(textstat_keyness(dfmat3, target = "2017-Trump"), 10)

# using the likelihood ratio method
head(textstat_keyness(dfm_smooth(dfmat3), measure = "lr", target = "2017-Trump"), 10)
```

`textstat_lexdiv` *Calculate lexical diversity*

Description

Calculate the lexical diversity of text(s).

Usage

```
textstat_lexdiv(
  x,
  measure = c("TTR", "C", "R", "CTTR", "U", "S", "K", "I", "D", "Vm", "Maas", "MATTR",
    "MSTTR", "all"),
  remove_numbers = TRUE,
  remove_punct = TRUE,
  remove_symbols = TRUE,
  remove_hyphens = FALSE,
  log.base = 10,
  MATTR_window = 100L,
  MSTTR_segment = 100L,
  ...
)
```

Arguments

`x`	an dfm or tokens input object for whose documents lexical diversity will be computed
`measure`	a character vector defining the measure to compute
`remove_numbers`	logical; if `TRUE` remove features or tokens that consist only of numerals (the Unicode "Number" `[N]` class)
`remove_punct`	logical; if `TRUE` remove all features or tokens that consist only of the Unicode "Punctuation" `[P]` class)
`remove_symbols`	logical; if `TRUE` remove all features or tokens that consist only of the Unicode "Punctuation" `[S]` class)
`remove_hyphens`	logical; if `TRUE` split words that are connected by hyphenation and hyphenation-like characters in between words, e.g. "self-storage" becomes two features or tokens "self" and "storage". Default is FALSE to preserve such words as is, with the hyphens.

log.base	a numeric value defining the base of the logarithm (for measures using logarithms)
MATTR_window	a numeric value defining the size of the moving window for computation of the Moving-Average Type-Token Ratio (Covington & McFall, 2010)
MSTTR_segment	a numeric value defining the size of the each segment for the computation of the the Mean Segmental Type-Token Ratio (Johnson, 1944)
...	not used directly

Details

textstat_lexdiv calculates the lexical diversity of documents using a variety of indices.

In the following formulas, N refers to the total number of tokens, V to the number of types, and $f_v(i, N)$ to the numbers of types occurring i times in a sample of length N.

"TTR": The ordinary *Type-Token Ratio*:

$$TTR = \frac{V}{N}$$

"C": Herdan's *C* (Herdan, 1960, as cited in Tweedie & Baayen, 1998; sometimes referred to as *LogTTR*):

$$C = \frac{\log V}{\log N}$$

"R": Guiraud's *Root TTR* (Guiraud, 1954, as cited in Tweedie & Baayen, 1998):

$$R = \frac{V}{\sqrt{N}}$$

"CTTR": Carroll's *Corrected TTR*:

$$CTTR = \frac{V}{\sqrt{2N}}$$

"U": Dugast's *Uber Index* (Dugast, 1978, as cited in Tweedie & Baayen, 1998):

$$U = \frac{(\log N)^2}{\log N - \log V}$$

"S": Summer's index:

$$S = \frac{\log \log V}{\log \log N}$$

"K": Yule's *K* (Yule, 1944, as presented in Tweedie & Baayen, 1998, Eq. 16) is calculated by:

$$K = 10^4 \times \left[-\frac{1}{N} + \sum_{i=1}^{V} f_v(i, N) \left(\frac{i}{N} \right)^2 \right]$$

"I": Yule's *I* (Yule, 1944) is calculated by:

$$I = \frac{V^2}{M_2 - V}$$

$$M_2 = \sum_{i=1}^{V} i^2 * f_v(i, N)$$

`"D"`: Simpson's *D* (Simpson 1949, as presented in Tweedie & Baayen, 1998, Eq. 17) is calculated by:

$$D = \sum_{i=1}^{V} f_v(i, N) \frac{i}{N} \frac{i-1}{N-1}$$

`"Vm"`: Herdan's V_m (Herdan 1955, as presented in Tweedie & Baayen, 1998, Eq. 18) is calculated by:

$$V_m = \sqrt{\sum_{i=1}^{V} f_v(i, N)(i/N)^2 - \frac{i}{V}}$$

`"Maas"`: Maas' indices (a, $\log V_0$ & $\log {}_e V_0$):

$$a^2 = \frac{\log N - \log V}{\log N^2}$$

$$\log V_0 = \frac{\log V}{\sqrt{1 - \frac{\log V}{\log N}^2}}$$

The measure was derived from a formula by Mueller (1969, as cited in Maas, 1972). $\log {}_e V_0$ is equivalent to $\log V_0$, only with e as the base for the logarithms. Also calculated are a, $\log V_0$ (both not the same as before) and V' as measures of relative vocabulary growth while the text progresses. To calculate these measures, the first half of the text and the full text will be examined (see Maas, 1972, p. 67 ff. for details). Note: for the current method (for a dfm) there is no computation on separate halves of the text.

`"MATTR"`: The Moving-Average Type-Token Ratio (Covington & McFall, 2010) calculates TTRs for a moving window of tokens from the first to the last token, computing a TTR for each window. The MATTR is the mean of the TTRs of each window.

`"MSTTR"`: Mean Segmental Type-Token Ratio (sometimes referred to as *Split TTR*) splits the tokens into segments of the given size, TTR for each segment is calculated and the mean of these values returned. When this value is < 1.0, it splits the tokens into equal, non-overlapping sections of that size. When this value is > 1, it defines the segments as windows of that size. Tokens at the end which do not make a full segment are ignored.

Value

A data.frame of documents and their lexical diversity scores.

Author(s)

Kenneth Benoit and Jiong Wei Lua. Many of the formulas have been reimplemented from functions written by Meik Michalke in the **koRpus** package.

References

Covington, M.A. & McFall, J.D. (2010). Cutting the Gordian Knot: The Moving-Average Type-Token Ratio (MATTR) *Journal of Quantitative Linguistics*, 17(2), 94–100. doi:10.1080/09296171003643098

Herdan, G. (1955). A New Derivation and Interpretation of Yule's 'Characteristic' *K*. *Zeitschrift für angewandte Mathematik und Physik*, 6(4): 332–334.

Maas, H.D. (1972). Über den Zusammenhang zwischen Wortschatzumfang und Länge eines Textes. *Zeitschrift für Literaturwissenschaft und Linguistik*, 2(8), 73–96.

McCarthy, P.M. & Jarvis, S. (2007). vocd: A Theoretical and Empirical Evaluation. *Language Testing*, 24(4), 459–488. doi:10.1177/0265532207080767

McCarthy, P.M. & Jarvis, S. (2010). MTLD, vocd-D, and HD-D: A Validation Study of Sophisticated Approaches to Lexical Diversity Assessment. *Behaviour Research Methods*, 42(2), 381–392.

Michalke, M. (2014). *koRpus: An R Package for Text Analysis (Version 0.05-4)*. Available from https://reaktanz.de/?c=hacking&s=koRpus.

Simpson, E.H. (1949). Measurement of Diversity. *Nature*, 163: 688. doi:10.1038/163688a0

Tweedie. F.J. and Baayen, R.H. (1998). How Variable May a Constant Be? Measures of Lexical Richness in Perspective. *Computers and the Humanities*, 32(5), 323–352. doi:10.1023/A:1001749303137

Yule, G. U. (1944) *The Statistical Study of Literary Vocabulary.* Cambridge: Cambridge University Press.

Examples

```
library("quanteda")

txt <- c("Anyway, like I was sayin', shrimp is the fruit of the sea. You can
          barbecue it, boil it, broil it, bake it, saute it.",
         "There's shrimp-kabobs,
          shrimp creole, shrimp gumbo. Pan fried, deep fried, stir-fried. There's
          pineapple shrimp, lemon shrimp, coconut shrimp, pepper shrimp, shrimp soup,
          shrimp stew, shrimp salad, shrimp and potatoes, shrimp burger, shrimp
          sandwich.")
tokens(txt) %>%
    textstat_lexdiv(measure = c("TTR", "CTTR", "K"))
dfm(tokens(txt)) %>%
    textstat_lexdiv(measure = c("TTR", "CTTR", "K"))

toks <- tokens(corpus_subset(data_corpus_inaugural, Year > 2000))
textstat_lexdiv(toks, c("CTTR", "TTR", "MATTR"), MATTR_window = 100)
```

---

textstat_readability *Calculate readability*

Description

Calculate the readability of text(s) using one of a variety of computed indexes.

Usage

```
textstat_readability(
  x,
  measure = "Flesch",
  remove_hyphens = TRUE,
  min_sentence_length = 1,
```

```
  max_sentence_length = 10000,
  intermediate = FALSE,
  ...
)
```

Arguments

`x`	a character or corpus object containing the texts
`measure`	character vector defining the readability measure to calculate. Matches are case-insensitive. See other valid measures under Details.
`remove_hyphens`	if `TRUE`, treat constituent words in hyphenated as separate terms, for purposes of computing word lengths, e.g. "decision-making" as two terms of lengths 8 and 6 characters respectively, rather than as a single word of 15 characters
`min_sentence_length`, `max_sentence_length`	set the minimum and maximum sentence lengths (in tokens, excluding punctuation) to include in the computation of readability. This makes it easy to exclude "sentences" that may not really be sentences, such as section titles, table elements, and other cruft that might be in the texts following conversion. For finer-grained control, consider filtering sentences prior first, including through pattern-matching, using corpus_trim().
`intermediate`	if `TRUE`, include intermediate quantities in the output
`...`	not used

Details

The following readability formulas have been implemented, where

- Nw = n_w = number of words
- Nc = n_c = number of characters
- Nst = n_{st} = number of sentences
- Nsy = n_{sy} = number of syllables
- Nwf = n_{wf} = number of words matching the Dale-Chall List of 3000 "familiar words"
- ASL = Average Sentence Length: number of words / number of sentences
- AWL = Average Word Length: number of characters / number of words
- AFW = Average Familiar Words: count of words matching the Dale-Chall list of 3000 "familiar words" / number of all words
- Nwd = n_{wd} = number of "difficult" words not matching the Dale-Chall list of "familiar" words

`"ARI"`: Automated Readability Index (Senter and Smith 1967)

$$0.5ASL + 4.71AWL - 21.34$$

`"ARI.Simple"`: A simplified version of Senter and Smith's (1967) Automated Readability Index.

$$ASL + 9AWL$$

`"Bormuth.MC"`: Bormuth's (1969) Mean Cloze Formula.

$$0.886593 - 0.03640 \times AWL + 0.161911 \times AFW - 0.21401 \times ASL - 0.000577 \times ASL^2 - 0.000005 \times ASL^3$$

`"Bormuth.GP"`: Bormuth's (1969) Grade Placement score.

$$4.275 + 12.881M - 34.934M^2 + 20.388M^3 + 26.194CCS - 2.046CCS^2 - 11.767CCS^3 - 42.285(M \times CCS) + 97.620($$

where M is the Bormuth Mean Cloze Formula as in `"Bormuth"` above, and CCS is the Cloze Criterion Score (Bormuth, 1968).

`"Coleman"`: Coleman's (1971) Readability Formula 1.

$$1.29 \times \frac{100 \times n_{wsy=1}}{n_w} - 38.45$$

where $n_{wsy=1}$ = Nwsy1 = the number of one-syllable words. The scaling by 100 in this and the other Coleman-derived measures arises because the Coleman measures are calculated on a per 100 words basis.

`"Coleman.C2"`: Coleman's (1971) Readability Formula 2.

$$1.16 \times \frac{100 \times n_{wsy=1}}{Nw + 1.48 \times \frac{100 \times n_{st}}{n_w} - 37.95}$$

`"Coleman.Liau.ECP"`: Coleman-Liau Estimated Cloze Percent (ECP) (Coleman and Liau 1975).

$$141.8401 - 0.214590 \times 100 \times AWL + 1.079812 \times \frac{n_{st} \times 100}{n_w}$$

`"Coleman.Liau.grade"`: Coleman-Liau Grade Level (Coleman and Liau 1975).

$$-27.4004 \times \texttt{Coleman.Liau.ECP} \times 100 + 23.06395$$

`"Coleman.Liau.short"`: Coleman-Liau Index (Coleman and Liau 1975).

$$5.88 \times AWL + 29.6 \times \frac{n_{st}}{n_w} - 15.8$$

`"Dale.Chall"`: The New Dale-Chall Readability formula (Chall and Dale 1995).

$$64 - (0.95 \times 100 \times \frac{n_{wd}}{n_w}) - (0.69 \times ASL)$$

`"Dale.Chall.Old"`: The original Dale-Chall Readability formula (Dale and Chall (1948).

$$0.1579 \times 100 \times \frac{n_{wd}}{n_w} + 0.0496 \times ASL [+3.6365]$$

The additional constant 3.6365 is only added if (Nwd / Nw) > 0.05.

`"Dale.Chall.PSK"`: The Powers-Sumner-Kearl Variation of the Dale and Chall Readability formula (Powers, Sumner and Kearl, 1958).

$$0.1155 \times 100 \frac{n_{wd}}{n_w}) + (0.0596 \times ASL) + 3.2672$$

`"Danielson.Bryan"`: Danielson-Bryan's (1963) Readability Measure 1.

$$(1.0364 \times \frac{n_c}{n_{blank}}) + (0.0194 \times \frac{n_c}{n_{st}}) - 0.6059$$

where n_{blank} = Nblank = the number of blanks.

`"Danielson.Bryan2"`: Danielson-Bryan's (1963) Readability Measure 2.

$$131.059 - (10.364 \times \frac{n_c}{n_{blank}}) + (0.0194 \times \frac{n_c}{n_{st}})$$

where n_{blank} = Nblank = the number of blanks.

`"Dickes.Steiwer"`: Dickes-Steiwer Index (Dicks and Steiwer 1977).

$$235.95993 - (7.3021 \times AWL) - (12.56438 \times ASL) - (50.03293 \times TTR)$$

where TTR is the Type-Token Ratio (see `textstat_lexdiv()`)

`"DRP"`: Degrees of Reading Power.

$$(1 - Bormuth.MC) * 100$$

where Bormuth.MC refers to Bormuth's (1969) Mean Cloze Formula (documented above)

`"ELF"`: Easy Listening Formula (Fang 1966):

$$\frac{n_{wsy>=2}}{n_{st}}$$

where $n_{wsy>=2}$ = Nwmin2sy = the number of words with 2 syllables or more.

`"Farr.Jenkins.Paterson"`: Farr-Jenkins-Paterson's Simplification of Flesch's Reading Ease Score (Farr, Jenkins and Paterson 1951).

$$-31.517 - (1.015 \times ASL) + (1.599 \times \frac{n_{wsy=1}}{n_w})$$

where $n_{wsy=1}$ = Nwsy1 = the number of one-syllable words.

`"Flesch"`: Flesch's Reading Ease Score (Flesch 1948).

$$206.835 - (1.015 \times ASL) - (84.6 \times \frac{n_{sy}}{n_w})$$

`"Flesch.PSK"`: The Powers-Sumner-Kearl's Variation of Flesch Reading Ease Score (Powers, Sumner and Kearl, 1958).

$$(0.0778 \times ASL) + (4.55 \times \frac{n_{sy}}{n_w}) - 2.2029$$

`"Flesch.Kincaid"`: Flesch-Kincaid Readability Score (Flesch and Kincaid 1975).

$$0.39 \times ASL + 11.8 \times \frac{n_{sy}}{n_w} - 15.59$$

`"FOG"`: Gunning's Fog Index (Gunning 1952).

$$0.4 \times (ASL + 100 \times \frac{n_{wsy>=3}}{n_w})$$

where $n_{wsy>=3}$ = Nwmin3sy = the number of words with 3-syllables or more. The scaling by 100 arises because the original FOG index is based on just a sample of 100 words)

`"FOG.PSK"`: The Powers-Sumner-Kearl Variation of Gunning's Fog Index (Powers, Sumner and Kearl, 1958).

$$3.0680 \times (0.0877 \times ASL) + (0.0984 \times 100 \times \frac{n_{wsy>=3}}{n_w})$$

where $n_{wsy>=3}$ = Nwmin3sy = the number of words with 3-syllables or more. The scaling by 100 arises because the original FOG index is based on just a sample of 100 words)

`"FOG.NRI"`: The Navy's Adaptation of Gunning's Fog Index (Kincaid, Fishburne, Rogers and Chissom 1975).

$$(\frac{(n_{wsy<3} + 3 \times n_{wsy=3})}{(100 \times \frac{N_{st}}{N_w})} - 3)/2$$

where $n_{wsy<3}$ = Nwless3sy = the number of words with *less than* 3 syllables, and $n_{wsy=3}$ = Nw3sy = the number of 3-syllable words. The scaling by 100 arises because the original FOG index is based on just a sample of 100 words)

`"FORCAST"`: FORCAST (Simplified Version of FORCAST.RGL) (Caylor and Sticht 1973).

$$20 - \frac{n_{wsy=1} \times 150)}{(n_w \times 10)}$$

where $n_{wsy=1}$ = Nwsy1 = the number of one-syllable words. The scaling by 150 arises because the original FORCAST index is based on just a sample of 150 words.

`"FORCAST.RGL"`: FORCAST.RGL (Caylor and Sticht 1973).

$$20.43 - 0.11 \times \frac{n_{wsy=1} \times 150)}{(n_w \times 10)}$$

where $n_{wsy=1}$ = Nwsy1 = the number of one-syllable words. The scaling by 150 arises because the original FORCAST index is based on just a sample of 150 words.

`"Fucks"`: Fucks' (1955) Stilcharakteristik (Style Characteristic).

$$AWL * ASL$$

`"Linsear.Write"`: Linsear Write (Klare 1975).

$$\frac{[(100 - (\frac{100 \times n_{wsy<3}}{n_w})) + (3 \times \frac{100 \times n_{wsy>=3}}{n_w})]}{(100 \times \frac{n_{st}}{n_w})}$$

where $n_{wsy<3}$ = Nwless3sy = the number of words with *less than* 3 syllables, and $n_{wsy>=3}$ = Nwmin3sy = the number of words with 3-syllables or more. The scaling by 100 arises because the original Linsear.Write measure is based on just a sample of 100 words)

`"LIW"`: Björnsson's (1968) Läsbarhetsindex (For Swedish Texts).

$$ASL + \frac{100 \times n_{wsy>=7}}{n_w}$$

where $n_{wsy>=7}$ = Nwmin7sy = the number of words with 7-syllables or more. The scaling by 100 arises because the Läsbarhetsindex index is based on just a sample of 100 words)

`"nWS"`: Neue Wiener Sachtextformeln 1 (Bamberger and Vanecek 1984).

$$19.35 \times \frac{n_{wsy>=3}}{n_w} + 0.1672 \times ASL + 12.97 \times \frac{b_{wchar>=6}}{n_w} - 3.27 \times \frac{n_{wsy=1}}{n_w} - 0.875$$

where $n_{wsy>=3}$ = Nwmin3sy = the number of words with 3 syllables or more, $n_{wchar>=6}$ = Nwmin6char = the number of words with 6 characters or more, and $n_{wsy=1}$ = Nwsy1 = the number of one-syllable words.

`"nWS.2"`: Neue Wiener Sachtextformeln 2 (Bamberger and Vanecek 1984).

$$20.07 \times \frac{n_{wsy>=3}}{n_w} + 0.1682 \times ASL + 13.73 \times \frac{n_{wchar>=6}}{n_w} - 2.779$$

where $n_{wsy>=3}$ = Nwmin3sy = the number of words with 3 syllables or more, and $n_{wchar>=6}$ = Nwmin6char = the number of words with 6 characters or more.

`"nWS.3"`: Neue Wiener Sachtextformeln 3 (Bamberger and Vanecek 1984).

$$29.63 \times \frac{n_{wsy>=3}}{n_w} + 0.1905 \times ASL - 1.1144$$

where $n_{wsy>=3}$ = Nwmin3sy = the number of words with 3 syllables or more.

`"nWS.4"`: Neue Wiener Sachtextformeln 4 (Bamberger and Vanecek 1984).

$$27.44 \times \frac{n_{wsy>=3}}{n_w} + 0.2656 \times ASL - 1.693$$

where $n_{wsy>=3}$ = Nwmin3sy = the number of words with 3 syllables or more.

`"RIX"`: Anderson's (1983) Readability Index.

$$\frac{n_{wsy>=7}}{n_{st}}$$

where $n_{wsy>=7}$ = Nwmin7sy = the number of words with 7-syllables or more.

`"Scrabble"`: Scrabble Measure.

$$MeanScrabbleLetterValuesofAllWords$$

. Scrabble values are for English. There is no reference for this, as we created it experimentally. It's not part of any accepted readability index!

`"SMOG"`: Simple Measure of Gobbledygook (SMOG) (McLaughlin 1969).

$$1.043 \times \sqrt{n_{wsy>=3}} \times \frac{30}{n_{st}} + 3.1291$$

where $n_{wsy>=3}$ = Nwmin3sy = the number of words with 3 syllables or more. This measure is regression equation D in McLaughlin's original paper.

`"SMOG.C"`: SMOG (Regression Equation C) (McLaughlin's 1969)

$$0.9986 \times \sqrt{Nwmin3sy \times \frac{30}{n_{st}} + 5} + 2.8795$$

where $n_{wsy>=3}$ = Nwmin3sy = the number of words with 3 syllables or more. This measure is regression equation C in McLaughlin's original paper.

`"SMOG.simple"`: Simplified Version of McLaughlin's (1969) SMOG Measure.

$$\sqrt{Nwmin3sy \times \frac{30}{n_{st}}} + 3$$

`"SMOG.de"`: Adaptation of McLaughlin's (1969) SMOG Measure for German Texts.

$$\sqrt{Nwmin3sy \times \frac{30}{n_{st}} - 2}$$

`"Spache"`: Spache's (1952) Readability Measure.

$$0.121 \times ASL + 0.082 \times \frac{n_{wnotinspache}}{n_w} + 0.659$$

where $n_{wnotinspache}$ = Nwnotinspache = number of unique words not in the Spache word list.

`"Spache.old"`: Spache's (1952) Readability Measure (Old).

$$0.141 \times ASL + 0.086 \times \frac{n_{wnotinspache}}{n_w} + 0.839$$

where $n_{wnotinspache}$ = Nwnotinspache = number of unique words not in the Spache word list.

`"Strain"`: Strain Index (Solomon 2006).

$$n_{sy} / \frac{n_{st}}{3} / 10$$

The scaling by 3 arises because the original Strain index is based on just the first 3 sentences.

`"Traenkle.Bailer"`: Tränkle & Bailer's (1984) Readability Measure 1.

$$224.6814 - (79.8304 \times AWL) - (12.24032 \times ASL) - (1.292857 \times 100 \times \frac{n_{prep}}{n_w}$$

where n_{prep} = Nprep = the number of prepositions. The scaling by 100 arises because the original Tränkle & Bailer index is based on just a sample of 100 words.

`"Traenkle.Bailer2"`: Tränkle & Bailer's (1984) Readability Measure 2.

$$Trnkle.Bailer2 = 234.1063 - (96.11069 \times AWL) - (2.05444 \times 100 \times \frac{n_{prep}}{n_w}) - (1.02805 \times 100 \times \frac{n_{conj}}{n_w}$$

where n_{prep} = Nprep = the number of prepositions, n_{conj} = Nconj = the number of conjunctions, The scaling by 100 arises because the original Tränkle & Bailer index is based on just a sample of 100 words)

`"Wheeler.Smith"`: Wheeler & Smith's (1954) Readability Measure.

$$ASL \times 10 \times \frac{n_{wsy>=2}}{n_{words}}$$

where $n_{wsy>=2}$ = Nwmin2sy = the number of words with 2 syllables or more.

`"meanSentenceLength"`: Average Sentence Length (ASL).

$$\frac{n_w}{n_{st}}$$

`"meanWordSyllables"`: Average Word Syllables (AWL).

$$\frac{n_{sy}}{n_w}$$

Value

`textstat_readability` returns a data.frame of documents and their readability scores.

Author(s)

Kenneth Benoit, re-engineered from Meik Michalke's **koRpus** package.

References

Anderson, J. (1983). Lix and rix: Variations on a little-known readability index. *Journal of Reading*, 26(6), 490–496. https://www.jstor.org/stable/40031755

Bamberger, R. & Vanecek, E. (1984). *Lesen-Verstehen-Lernen-Schreiben*. Wien: Jugend und Volk.

Björnsson, C. H. (1968). *Läsbarhet*. Stockholm: Liber.

Bormuth, J.R. (1969). Development of Readability Analysis.

Bormuth, J.R. (1968). Cloze test readability: Criterion reference scores. *Journal of educational measurement*, 5(3), 189–196. https://www.jstor.org/stable/1433978

Caylor, J.S. (1973). Methodologies for Determining Reading Requirements of Military Occupational Specialities. https://eric.ed.gov/?id=ED074343

Caylor, J.S. & Sticht, T.G. (1973). *Development of a Simple Readability Index for Job Reading Material* https://archive.org/details/ERIC_ED076707

Coleman, E.B. (1971). Developing a technology of written instruction: Some determiners of the complexity of prose. *Verbal learning research and the technology of written instruction*, 155–204.

Coleman, M. & Liau, T.L. (1975). A Computer Readability Formula Designed for Machine Scoring. *Journal of Applied Psychology*, 60(2), 283. doi:10.1037/h0076540

Dale, E. and Chall, J.S. (1948). A Formula for Predicting Readability: Instructions. *Educational Research Bulletin*, 37-54. https://www.jstor.org/stable/1473169

Chall, J.S. and Dale, E. (1995). *Readability Revisited: The New Dale-Chall Readability Formula*. Brookline Books.

Dickes, P. & Steiwer, L. (1977). Ausarbeitung von Lesbarkeitsformeln für die Deutsche Sprache. *Zeitschrift für Entwicklungspsychologie und Pädagogische Psychologie* 9(1), 20–28.

Danielson, W.A., & Bryan, S.D. (1963). Computer Automation of Two Readability Formulas. *Journalism Quarterly*, 40(2), 201–206. doi:10.1177/107769906304000207

DuBay, W.H. (2004). *The Principles of Readability*.

Fang, I. E. (1966). The "Easy listening formula". *Journal of Broadcasting & Electronic Media*, 11(1), 63–68. doi:10.1080/08838156609363529

Farr, J. N., Jenkins, J.J., & Paterson, D.G. (1951). Simplification of Flesch Reading Ease Formula. *Journal of Applied Psychology*, 35(5): 333. doi:10.1037/h0057532

Flesch, R. (1948). A New Readability Yardstick. *Journal of Applied Psychology*, 32(3), 221. doi:10.1037/h0057532

Fucks, W. (1955). Der Unterschied des Prosastils von Dichtern und anderen Schriftstellern. *Sprachforum*, 1, 233-244.

Gunning, R. (1952). *The Technique of Clear Writing*. New York: McGraw-Hill.

Klare, G.R. (1975). Assessing Readability. *Reading Research Quarterly*, 10(1), 62-102. doi:10.2307/747086

Kincaid, J. P., Fishburne Jr, R.P., Rogers, R.L., & Chissom, B.S. (1975). Derivation of New Readability Formulas (Automated Readability Index, FOG count and Flesch Reading Ease Formula) for Navy Enlisted Personnel.

McLaughlin, G.H. (1969). SMOG Grading: A New Readability Formula. *Journal of Reading*, 12(8), 639-646.

Michalke, M. (2014). *koRpus: An R Package for Text Analysis (Version 0.05-4)*. Available from https://reaktanz.de/?c=hacking&s=koRpus.

Powers, R.D., Sumner, W.A., and Kearl, B.E. (1958). A Recalculation of Four Adult Readability Formulas. *Journal of Educational Psychology*, 49(2), 99. doi:10.1037/h0043254

Senter, R. J., & Smith, E. A. (1967). Automated readability index. Wright-Patterson Air Force Base. Report No. AMRL-TR-6620.

*Solomon, N. W. (2006). *Qualitative Analysis of Media Language*. India.

Spache, G. (1953). "A new readability formula for primary-grade reading materials." *The Elementary School Journal*, 53, 410–413. https://www.jstor.org/stable/998915

Tränkle, U. & Bailer, H. (1984). Kreuzvalidierung und Neuberechnung von Lesbarkeitsformeln für die deutsche Sprache. *Zeitschrift für Entwicklungspsychologie und Pädagogische Psychologie*, 16(3), 231–244.

Wheeler, L.R. & Smith, E.H. (1954). A Practical Readability Formula for the Classroom Teacher in the Primary Grades. *Elementary English*, 31, 397–399. https://www.jstor.org/stable/41384251

*Nimaldasan is the pen name of N. Watson Solomon, Assistant Professor of Journalism, School of Media Studies, SRM University, India.

Examples

```
txt <- c(doc1 = "Readability zero one. Ten, Eleven.",
         doc2 = "The cat in a dilapidated tophat.")
textstat_readability(txt, measure = "Flesch")
textstat_readability(txt, measure = c("FOG", "FOG.PSK", "FOG.NRI"))

textstat_readability(quanteda::data_corpus_inaugural[48:58],
                     measure = c("Flesch.Kincaid", "Dale.Chall.old"))
```

`textstat_simil` *Similarity and distance computation between documents or features*

Description

These functions compute matrixes of distances and similarities between documents or features from a dfm and return a matrix of similarities or distances in a sparse format. These methods are fast and robust because they operate directly on the sparse dfm objects. The output can easily be coerced to an ordinary matrix, a data.frame of pairwise comparisons, or a dist format.

Usage

```
textstat_simil(
  x,
  y = NULL,
  selection = NULL,
  margin = c("documents", "features"),
 method = c("correlation", "cosine", "jaccard", "ejaccard", "dice", "edice", "hamann",
    "simple matching"),
  min_simil = NULL,
  ...
)

textstat_dist(
  x,
  y = NULL,
  selection = NULL,
  margin = c("documents", "features"),
  method = c("euclidean", "manhattan", "maximum", "canberra", "minkowski"),
  p = 2,
  ...
)
```

Arguments

x, y	a dfm objects; y is an optional target matrix matching x in the margin on which the similarity or distance will be computed.
selection	(deprecated - use y instead).
margin	identifies the margin of the dfm on which similarity or difference will be computed: "documents" for documents or "features" for word/term features.
method	character; the method identifying the similarity or distance measure to be used; see Details.
min_simil	numeric; a threshold for the similarity values below which similarity values will not be returned
...	unused
p	The power of the Minkowski distance.

Details

`textstat_simil` options are: `"correlation"` (default), `"cosine"`, `"jaccard"`, `"ejaccard"`, `"dice"`, `"edice"`, `"simple matching"`, and `"hamann"`.

`textstat_dist` options are: `"euclidean"` (default), `"manhattan"`, `"maximum"`, `"canberra"`, and `"minkowski"`.

Value

A sparse matrix from the **Matrix** package that will be symmetric unless y is specified.

Conversion to other data types

The output objects from `textstat_simil()` and `textstat_dist()` can be transformed easily into a list format using `as.list()`, which returns a list for each unique element of the second of the pairs, a data.frame using `as.data.frame()`, which returns pairwise scores, `as.dist()`for a dist object, or `as.matrix()` to convert it into an ordinary matrix.

Note

If you want to compute similarity on a "normalized" dfm object (controlling for variable document lengths, for methods such as correlation for which different document lengths matter), then wrap the input dfm in `[dfm_weight](x, "prop")`.

See Also

`as.list.textstat_proxy()`, `as.data.frame.textstat_proxy()`, `stats::as.dist()`

Examples

```
# similarities for documents
library("quanteda")
dfmat <- corpus_subset(data_corpus_inaugural, Year > 2000) %>%
    tokens(remove_punct = TRUE) %>%
    tokens_remove(stopwords("english")) %>%
    dfm()
(tstat1 <- textstat_simil(dfmat, method = "cosine", margin = "documents"))
as.matrix(tstat1)
as.list(tstat1)
as.list(tstat1, diag = TRUE)

# min_simil
(tstat2 <- textstat_simil(dfmat, method = "cosine", margin = "documents", min_simil = 0.6))
as.matrix(tstat2)

# similarities for for specific documents
textstat_simil(dfmat, dfmat["2017-Trump", ], margin = "documents")
textstat_simil(dfmat, dfmat["2017-Trump", ], method = "cosine", margin = "documents")
textstat_simil(dfmat, dfmat[c("2009-Obama", "2013-Obama"), ], margin = "documents")

# compute some term similarities
```

```
tstat3 <- textstat_simil(dfmat, dfmat[, c("fair", "health", "terror")], method = "cosine",
                         margin = "features")
head(as.matrix(tstat3), 10)
as.list(tstat3, n = 6)


# distances for documents
(tstat4 <- textstat_dist(dfmat, margin = "documents"))
as.matrix(tstat4)
as.list(tstat4)
as.dist(tstat4)

# distances for specific documents
textstat_dist(dfmat, dfmat["2017-Trump", ], margin = "documents")
(tstat5 <- textstat_dist(dfmat, dfmat[c("2009-Obama" , "2013-Obama"), ], margin = "documents"))
as.matrix(tstat5)
as.list(tstat5)

## Not run:
# plot a dendrogram after converting the object into distances
plot(hclust(as.dist(tstat4)))

## End(Not run)
```

`textstat_summary` *Summarize documents as syntactic and lexical feature counts*

Description

Count syntactic and lexical features of documents such as tokens, types, sentences, and character categories.

Usage

```
textstat_summary(x, ...)
```

Arguments

x	corpus to be summarized
...	additional arguments passed through to dfm()

Details

Count the total number of characters, tokens and sentences as well as special tokens such as numbers, punctuation marks, symbols, tags and emojis.

- chars = number of characters; equal to `nchar()`
- sents = number of sentences; equal `ntoken(tokens(x), what = "sentence")`
- tokens = number of tokens; equal to ntoken()

- types = number of unique tokens; equal to ntype()
- puncts = number of punctuation marks (`^\p{P}+$`)
- numbers = number of numeric tokens (`^\p{Sc}{0,1}\p{N}+([.,]*\p{N})*\p{Sc}{0,1}$`)
- symbols = number of symbols (`^\p{S}$`)
- tags = number of tags; sum of `pattern_username` and `pattern_hashtag` in `quanteda::quanteda_options()`
- emojis = number of emojis (`^\p{Emoji_Presentation}+$`)

Examples

```
if (Sys.info()["sysname"] != "SunOS") {
library("quanteda")
corp <- data_corpus_inaugural[1:5]
textstat_summary(corp)
toks <- tokens(corp)
textstat_summary(toks)
dfmat <- dfm(toks)
textstat_summary(dfmat)
}
```

Index

∗ **collocations**
    textstat_collocations, 3
∗ **datasets**
    data_char_wordlists, 2
∗ **plot**
    textstat_frequency, 6
∗ **textstat**
    textstat_collocations, 3
    textstat_keyness, 8
    textstat_summary, 24

as.data.frame(), *23*
as.data.frame.textstat_proxy(), *23*
as.list(), *23*
as.list.textstat_proxy(), *23*

base::rank(), *6*

corpus, *3*, *14*
corpus_trim(), *14*

data_char_wordlists, 2
dfm, *6*, *8*, *10*, *22*
dfm(), *24*
dfm_group(), *6*
dist, *22*, *23*
docvars, *6*

nchar(), *24*
ntoken(), *24*
ntype(), *25*

quanteda::quanteda_options(), *25*

stats::as.dist(), *23*

textstat_collocations, 3
textstat_dist *(*textstat_simil*)*, 22
textstat_entropy, 5
textstat_frequency, 6
textstat_keyness, 8
textstat_lexdiv, 10
textstat_lexdiv(), *16*
textstat_readability, 13
textstat_simil, 22
textstat_summary, 24
tokens, *3*, *4*, *10*
tokens(), *4*