Package 'sts'

October 23, 2026

Type Package

Title Estimation of the Structural Topic and Sentiment-Discourse Model for Text Analysis

Version 1.6

Date 2026-10-23

Description The Structural Topic and Sentiment-Discourse (STS) model allows researchers to estimate topic models with document-level metadata that determines both topic prevalence and sentiment-discourse. The sentiment-discourse is modeled as a document-level latent variable for each topic that modulates the word frequency within a topic. These latent topic sentiment-discourse variables are controlled by the document-level metadata. The STS model can be useful for regression analysis with text data in addition to topic modeling's traditional use of descriptive analysis. The method was developed in Chen and Mankad (2025) <doi:10.1287/mnsc.2022.00261>.

License MIT + file LICENSE

URL https://github.com/mankad-research/sts_r

BugReports https://github.com/mankad-research/sts_r/issues

Depends R (>= 3.5.0)

Imports Rcpp, glmnet, matrixStats, slam, foreach, doParallel, parallel, Matrix, mvtnorm, ggplot2, tm, SnowballC, graphics, stats

Suggests R.rsp

LinkingTo Rcpp, RcppArmadillo

VignetteBuilder R.rsp

Encoding UTF-8

LazyData true

RoxygenNote 7.3.1

NeedsCompilation yes

Author Shawn Mankad [aut, cre], Li Chen [aut], Margaret E. Roberts [ctb, cph] (Author of adapted code from the stm

package),
Brandon M. Stewart [ctb, cph] (Author of adapted code from the stm package),
Dustin Tingley [ctb, cph] (Author of adapted code from the stm package)

Maintainer Shawn Mankad <smankad@ncsu.edu>

Repository CRAN

Date/Publication 2026-10-23 15:10:02 UTC

Contents

sts-package . . . 2
estimateRegns . . . 3
findRepresentativeDocs . . . 4
gadarian . . . 5
heldoutLikelihood . . . 6
make.heldout . . . 7
plot.STS . . . 8
plotRepresentativeDocs . . . 9
prepDocuments . . . 10
printRegnTables . . . 11
printTopWords . . . 12
sts . . . 13
summary.STS . . . 16
textProcessor . . . 16
topicExclusivity . . . 19
topicSemanticCoherence . . . 20

Index **22**

`sts-package` *A Structural Topic and Sentiment-Discourse Model for Text Analysis*

Description

This package implements the Structural Topic and Sentiment-Discourse (STS) model, which allows researchers to estimate topic models with document-level metadata that determines both topic prevalence and sentiment-discourse. The sentiment-discourse is modeled as a document-level latent variable for each topic that modulates the word frequency within a topic. These latent topic sentiment-discourse variables are controlled by the document-level metadata. The STS model can be useful for regression analysis with text data in addition to topic modeling's traditional use of descriptive analysis.

Details

Function to fit the model: `sts`

Functions for preparing text: `textProcessor` `prepDocuments` `make.heldout`

Functions for Post-Estimation: `estimateRegns` `topicExclusivity` `topicSemanticCoherence` `heldoutLikelihood` `plotRepresentativeDocs` `findRepresentativeDocs` `printTopWords` `plot.STS`

Since version 1.6 the package does not depend on the stm package: the spectral (anchor word) initialization, the STM fit used by `initialization = "stm"`, and the text-preparation helpers are implemented internally, adapted from stm (Roberts, Stewart and Tingley; MIT license).

Author(s)

Author: Shawn Mankad and Li Chen

Maintainer: Shawn Mankad smankad@ncsu.edu

References

Chen, L. and Mankad, S. (2025) "A Structural Topic and Sentiment-Discourse Model for Text Analysis" Management Science, 71(7), pp.5767-5787.

See Also

`sts`

`estimateRegns` *Regression Table Estimation*

Description

Estimates regression tables for prevalence and sentiment/discourse.

Usage

```
estimateRegns(object, prevalence_sentiment, corpus)
```

Arguments

`object`	an sts object
`prevalence_sentiment`	A formula object with no response variable or a design matrix with the covariates. If a formula, the variables must be contained in corpus$meta.
`corpus`	The document term matrix to be modeled in a sparse term count matrix with one row per document and one column per term. The object must be a list of with each element corresponding to a document. Each document is represented as an integer matrix with two rows, and columns equal to the number of unique vocabulary words in the document. The first row contains the 1-indexed vocabulary entry and the second row contains the number of times that term appears. This is the format produced by `prepDocuments`.

Details

Estimate Gamma coefficients (along with standard errors, p-values, etc.) to assess how document-level meta-data determine prevalence and sentiment/discourse

Value

a list of tables with regression coefficient estimates. The first num-topic elements pertain to prevalence; the latter num-topic elements pertain to sentiment-discourse.

Examples

```
library("sts")
temp<-textProcessor(documents=gadarian$open.ended.response,
metadata=gadarian, verbose = FALSE)
out <- prepDocuments(temp$documents, temp$vocab, temp$meta, verbose = FALSE)
out$meta$noTreatment <- ifelse(out$meta$treatment == 1, -1, 1)
## low max iteration number just for testing
sts_estimate <- sts(~ treatment*pid_rep, ~ noTreatment, out, K = 3, maxIter = 2)
regns <- estimateRegns(sts_estimate, ~treatment*pid_rep, out)
printRegnTables(x = regns)
```

---

findRepresentativeDocs

Function for Identifying Documents that Load Heavily on a Topic

---

Description

Extracts documents with the highest prevalence for a given topic

Usage

```
findRepresentativeDocs(object, corpus_text, topic, n = 3)
```

Arguments

object	Model output from sts
corpus_text	vector of text documents, usually contained in the output of prepDocuments
topic	a single topic number
n	number of documents to extract

Examples

```
#Examples with the Gadarian Data
library("sts")
temp<-textProcessor(documents=gadarian$open.ended.response,
metadata=gadarian, verbose = FALSE)
out <- prepDocuments(temp$documents, temp$vocab, temp$meta, verbose = FALSE)
out$meta$noTreatment <- ifelse(out$meta$treatment == 1, -1, 1)
## low max iteration number just for testing
sts_estimate <- sts(~ treatment*pid_rep, ~ noTreatment, out, K = 3, maxIter = 2)
docs <- findRepresentativeDocs(sts_estimate, out$meta$open.ended.response, topic = 3, n = 4)
plotRepresentativeDocs(docs, text.cex = 0.7, width = 100)
```

`gadarian` *Gadarian and Albertson data*

Description

This data set contains variables from Gadarian and Albertson (2014). The experiment had those in the treatment condition write about what made them anxious about immigration. The control condition just had subjects write about immigration. The data set is also distributed with the stm package (Roberts, Stewart and Tingley; MIT license) and is included here so that the examples do not require stm.

Usage

```
gadarian
```

Format

A data frame with 341 observations on the following 4 variables.

`MetaID` A numeric vector containing identification numbers; not used for analysis

`treatment` A numeric vector indicating treatment condition

`pid_rep` A numeric vector of party identification

`open.ended.response` A character vector of the subject's open ended response

Source

Gadarian, Shana Kushner, and Bethany Albertson. "Anxiety, immigration, and the search for information." Political Psychology 35.2 (2014): 133-164.

Roberts, Margaret E., Brandon M. Stewart, Dustin Tingley, Christopher Lucas, Jetson Leder-Luis, Shana Kushner Gadarian, Bethany Albertson, and David G. Rand. "Structural Topic Models for Open-Ended Survey Responses." American Journal of Political Science 58, no 4 (2014): 1064-1082.

Examples

```
head(gadarian)
temp <- textProcessor(documents = gadarian$open.ended.response,
                      metadata = gadarian, verbose = FALSE)
out <- prepDocuments(temp$documents, temp$vocab, temp$meta)
```

`heldoutLikelihood` *Compute Heldout Log-Likelihood*

Description

Compute the heldout log-likelihood of the STS model

Usage

```
heldoutLikelihood(object, missing)
```

Arguments

`object`	an sts object, typically estimated on a corpus created with `make.heldout`
`missing`	list of which words and documents are in the heldout set

Value

expected.heldout is the average of the held-out log-likelihood values for each document.

Examples

```
library("sts")
temp<-textProcessor(documents=gadarian$open.ended.response,
metadata=gadarian, verbose = FALSE)
out <- prepDocuments(temp$documents, temp$vocab, temp$meta, verbose = FALSE)
out$meta$noTreatment <- ifelse(out$meta$treatment == 1, -1, 1)
out_ho <- make.heldout(out$documents, out$vocab, seed = 1)
out_ho$meta <- out$meta
## low max iteration number just for testing
sts_estimate <- sts(~ treatment*pid_rep, ~ noTreatment, out_ho, K = 3, maxIter = 2, verbose = FALSE)
heldoutLikelihood(sts_estimate, out_ho$missing)$expected.heldout
```

`make.heldout` *Heldout Likelihood by Document Completion*

Description

Tools for making and evaluating heldout datasets. It is a port of the function of the same name in the stm package, so that heldout data can be created for `heldoutLikelihood` without installing stm.

Usage

```
make.heldout(
  documents,
  vocab,
  N = floor(0.1 * length(documents)),
  proportion = 0.5,
  seed = NULL
)
```

Arguments

`documents`	the documents to be modeled (see `sts` for format).
`vocab`	the vocabulary item
`N`	number of docs to be partially held out
`proportion`	proportion of docs to be held out.
`seed`	the seed, set for replicability

Details

These functions are used to create and evaluate heldout likelihood using the document completion method. The basic idea is to hold out some fraction of the words in a set of documents, train the model and use the document-level latent variables to evaluate the probability of the heldout portion. See the example for the basic workflow.

Value

A list with elements `documents` and `vocab` (the training corpus) and `missing`, a list with the indices of the partially held-out documents (`index`) and their held-out words (`docs`).

See Also

`heldoutLikelihood`

Examples

```
temp <- textProcessor(documents = gadarian$open.ended.response,
                      metadata = gadarian, verbose = FALSE)
out <- prepDocuments(temp$documents, temp$vocab, temp$meta, verbose = FALSE)
out_ho <- make.heldout(out$documents, out$vocab, seed = 1)
length(out_ho$missing$index)
```

---

plot.STS *Function for plotting STS objects*

Description

Produces a plot of the most likely words and their probabilities for each topic for different levels of sentiment for an STS object.

Usage

```
## S3 method for class 'STS'
plot(
  x,
  n = 10,
  topics = NULL,
  lowerPercentile = 0.05,
  upperPercentile = 0.95,
  ...
)
```

Arguments

x	Model output from sts.
n	Sets the number of words used to label each topic. In perspective plots it approximately sets the total number of words in the plot. n must be greater than or equal to 2
topics	Vector of topics to display. Defaults to all topics.
lowerPercentile	Percentile to calculate a representative negative sentiment document.
upperPercentile	Percentile to calculate a representative positive sentiment document.
...	Additional parameters passed to plotting functions.

Examples

```
#Examples with the Gadarian Data
library("sts")
temp<-textProcessor(documents=gadarian$open.ended.response,
metadata=gadarian, verbose = FALSE)
```

```
out <- prepDocuments(temp$documents, temp$vocab, temp$meta, verbose = FALSE)
out$meta$noTreatment <- ifelse(out$meta$treatment == 1, -1, 1)
## low max iteration number just for testing
sts_estimate <- sts(~ treatment*pid_rep, ~ noTreatment, out, K = 3, maxIter = 2)
plot(sts_estimate)
plot(sts_estimate, n = 10, topic = c(1,2))
```

`plotRepresentativeDocs` *Plot Documents that Load Heavily on a Topic*

Description

Produces a plot of the text of documents that load most heavily on topics for an STS object

Usage

```
plotRepresentativeDocs(object, text.cex = 1, width = 100)
```

Arguments

`object`	Model output from sts.
`text.cex`	Size of the text; Defaults to 1
`width`	Size of the plotting window; Defaults to 100

Examples

```
#Examples with the Gadarian Data
library("sts")
temp<-textProcessor(documents=gadarian$open.ended.response,
metadata=gadarian, verbose = FALSE)
out <- prepDocuments(temp$documents, temp$vocab, temp$meta, verbose = FALSE)
out$meta$noTreatment <- ifelse(out$meta$treatment == 1, -1, 1)
## low max iteration number just for testing
sts_estimate <- sts(~ treatment*pid_rep, ~ noTreatment, out, K = 3, maxIter = 2)
docs <- findRepresentativeDocs(sts_estimate, out$meta$open.ended.response, topic = 3, n = 1)
plotRepresentativeDocs(docs, text.cex = 0.7, width = 100)
```

prepDocuments *Prepare Documents for Analysis*

Description

Performs several corpus manipulations including removing words and renumbering word indices (to correct for zero-indexing and/or unused words in the vocab vector). It is a port of the function of the same name in the stm package, so that a corpus can be prepared for `sts` without installing stm.

Usage

```
prepDocuments(
  documents,
  vocab,
  meta = NULL,
  lower.thresh = 1,
  upper.thresh = Inf,
  subsample = NULL,
  verbose = TRUE
)
```

Arguments

`documents`	List of documents. For more on the format see `sts`.
`vocab`	Character vector of words in the vocabulary.
`meta`	Document metadata.
`lower.thresh`	Words which do not appear in a number of documents greater than lower.thresh will be dropped and both the documents and vocab files will be renumbered accordingly. If this causes all words within a document to be dropped, a message will print to the screen and it will also return vector of the documents removed so you can update your meta data as well. See details below.
`upper.thresh`	As with lower.thresh but this provides an upper bound. Words which appear in at least this number of documents will be dropped. Defaults to `Inf` which does no filtering.
`subsample`	If an integer will randomly subsample (without replacement) the given number of documents from the total corpus before any processing. Defaults to `NULL` which provides no subsampling. Note that the output may have fewer than the number of requested documents if additional processing causes some of those documents to be dropped.
`verbose`	A logical indicating whether or not to print details to the screen.

Details

The default setting `lower.thresh=1` means that words which appear in only one document will be dropped. This is often advantageous as there is little information about these words but the added

cost of including them in the model can be quite large. In many cases it will be helpful to set this threshold considerably higher. If the vocabulary is in excess of 5000 entries inference can slow quite a bit.

If words are removed, the function returns a vector of the original indices for the dropped items. If it removed documents it returns a vector of doc indices removed. Users with accompanying metadata or texts may want to drop those rows from the corresponding objects.

If you have a large corpus, consider setting `verbose = FALSE`.

Value

A list containing a new documents and vocab object.

`documents`	The new documents object for use with `sts`
`vocab`	The new vocab object
`meta`	The new meta data object
`docs.removed`	A set of indices corresponding to documents that were removed, if any.
`words.removed`	A set of indices corresponding to words that were removed, if any
`tokens.removed`	The total number of tokens removed
`wordcounts`	A table giving the the number of documents that each word is found in of the original document set, prior to any removal. This can be passed through a histogram for visual inspection

See Also

`textProcessor`

Examples

```
temp <- textProcessor(documents = gadarian$open.ended.response,
                      metadata = gadarian, verbose = FALSE)
out <- prepDocuments(temp$documents, temp$vocab, temp$meta)
length(out$vocab)
```

---

printRegnTables *Print Estimated Regression Tables*

Description

Prints estimated regression tables from estimateRegnTables()

Usage

```
printRegnTables(
  x,
  topics = NULL,
  digits = max(3L, getOption("digits") - 3L),
  signif.stars = getOption("show.signif.stars"),
  ...
)
```

Arguments

x	the estimated regression tables from estimateRegnTables()
topics	Vector of topics to display. Defaults to all topics.
digits	minimum number of significant digits to be used for most numbers.
signif.stars	logical; if TRUE, P-values are additionally encoded visually as 'significance stars' in order to help scanning of long coefficient tables. It defaults to the show.signif.stars slot of options.
...	other arguments suitable for stats::printCoefmat()

Value

Prints estimated regression tables from estimateRegnTables() to console

Examples

```
library("sts")
temp<-textProcessor(documents=gadarian$open.ended.response,
metadata=gadarian, verbose = FALSE)
out <- prepDocuments(temp$documents, temp$vocab, temp$meta, verbose = FALSE)
out$meta$noTreatment <- ifelse(out$meta$treatment == 1, -1, 1)
## low max iteration number just for testing
sts_estimate <- sts(~ treatment*pid_rep, ~ noTreatment, out, K = 3, maxIter = 2)
regns <- estimateRegns(sts_estimate, ~treatment*pid_rep, out)
printRegnTables(x = regns)
```

---

printTopWords *Print Top Words that Load Heavily on each Topic*

Description

Prints the top words for each document for low, average, and high levels of sentiment-discourse

Usage

```
printTopWords(object, n = 10, lowerPercentile = 0.05, upperPercentile = 0.95)
```

Arguments

object	Model output from sts
n	number of words to print to console for each topic
lowerPercentile	Percentile to calculate a representative negative sentiment document.
upperPercentile	Percentile to calculate a representative positive sentiment document.

Examples

```
#Examples with the Gadarian Data
library("sts")
temp<-textProcessor(documents=gadarian$open.ended.response,
metadata=gadarian, verbose = FALSE)
out <- prepDocuments(temp$documents, temp$vocab, temp$meta, verbose = FALSE)
out$meta$noTreatment <- ifelse(out$meta$treatment == 1, -1, 1)
## low max iteration number just for testing
sts_estimate <- sts(~ treatment*pid_rep, ~ noTreatment, out, K = 3, maxIter = 2)
printTopWords(sts_estimate)
```

---

sts *Variational EM for the Structural Topic and Sentiment-Discourse (STS) Model*

Description

Estimation of the STS Model using variational EM. The function takes sparse representation of a document-term matrix, covariates for each document, and an integer number of topics and returns fitted model parameters. See an overview of functions in the package here: sts-package

Usage

```
sts(
  prevalence_sentiment,
  initializationVar,
  corpus,
  K,
  maxIter = 100,
  convTol = 1e-05,
  initialization = "anchor",
  kappaEstimation = "lasso",
  verbose = TRUE,
  parallelize = FALSE,
  stmSeed = NULL
)
```

Arguments

prevalence_sentiment	A formula object with no response variable or a design matrix with the covariates. The variables must be contained in corpus$meta.
initializationVar	A formula with a single variable for use in the initialization of latent sentiment. This argument is usually the key experimental variable (e.g., review rating binary indicator of experiment/control group).
corpus	The document term matrix to be modeled in a sparse term count matrix with one row per document and one column per term. The object must be a list of with each element corresponding to a document. Each document is represented as an integer matrix with two rows, and columns equal to the number of unique vocabulary words in the document. The first row contains the 1-indexed vocabulary entry and the second row contains the number of times that term appears. This is the format produced by prepDocuments (and used by the stm package).
K	A positive integer (of size 2 or greater) representing the desired number of topics.
maxIter	A positive integer representing the max number of VEM iterations allowed.
convTol	Convergence tolerance for the variational EM estimation algorithm; Default value = 1e-5.
initialization	Character argument that allows the user to specify an initialization method. The default choice, "anchor" to initialize prevalence according to anchor words and the key experimental covariate identified in argument initializationVar. One can also use "stm", which uses a fitted STM model (Roberts et al. 2014, 2016) to initialize coefficients related to prevalence and sentiment-discourse. The STM model is fit internally (with initializationVar as the content covariate), so the stm package does not need to be installed. Its topic proportions initialize the prevalence variables, and its content-covariate word effects initialize kappa: for each topic, the sentiment-discourse coefficients are the slope (and the topic coefficients the intercept) of a document-weighted least-squares fit of the STM content effects on the centered levels of initializationVar, which reproduces the STM topic-word distributions exactly when initializationVar has two levels.
kappaEstimation	A character input specifying how kappa should be estimated. "lasso" (default) allows for penalties on the L1 norm. We estimate a regularization path and then select the optimal shrinkage parameter using AIC. "adjusted" utilizes the lasso penalty with an adjusted aggregated Poisson regression. All options use an approximation framework developed in Taddy (2013) called Distributed Multinomial Regression which utilizes a factorized poisson approximation to the multinomial. See Chen and Mankad (2025) on the implementation here.
verbose	A logical flag indicating whether information should be printed to the screen.
parallelize	A logical flag indicating whether to parallelize the estimation using all but one CPU cores on your local machine.
stmSeed	A prefit STM model object (e.g., from stm::stm() with initializationVar as the content covariate) to initialize the STS model. Note this is ignored unless initialization = "stm"

Details

This is the main function for estimating the Structural Topic and Sentiment-Discourse (STS) Model. Users provide a corpus of documents and a number of topics. Each word in a document comes from exactly one topic and each document is represented by the proportion of its words that come from each of the topics. The document-specific content covariates affect how much (prevalence) and the way in which a topic is discussed (sentiment-discourse).

Value

An object of class sts

`alpha`	Estimated prevalence and sentiment-discourse values for each document and topic
`gamma`	Estimated regression coefficients that determine prevalence and sentiment/discourse for each topic
`kappa`	Estimated kappa coefficients that determine sentiment-discourse and the topic-word distributions
`sigma_inv`	Inverse of the covariance matrix for the alpha parameters
`sigma`	Covariance matrix for the alpha parameters
`elbo`	the ELBO at each iteration of the estimation algorithm
`mv`	the baseline log-transformed occurrence rate of each word in the corpus
`runtime`	Time elapsed in seconds
`vocab`	Vocabulary vector used
`mu`	Mean (fitted) values for alpha based on document-level variables * estimated Gamma for each document

References

Roberts, M., Stewart, B., Tingley, D., and Airoldi, E. (2013) "The structural topic model and applied social science." In Advances in Neural Information Processing Systems Workshop on Topic Models: Computation, Application, and Evaluation.

Roberts M., Stewart, B. and Airoldi, E. (2016) "A model of text for experimentation in the social sciences" Journal of the American Statistical Association.

Chen, L. and Mankad, S. (2025) "A Structural Topic and Sentiment-Discourse Model for Text Analysis" Management Science, 71(7), pp.5767-5787.

See Also

`estimateRegns`

Examples

```
#An example using the Gadarian data.  From raw text to
# fitted model using textProcessor() which leverages the tm Package
library("sts")
temp<-textProcessor(documents=gadarian$open.ended.response,
```

```
metadata=gadarian, verbose = FALSE)
out <- prepDocuments(temp$documents, temp$vocab, temp$meta, verbose = FALSE)
out$meta$noTreatment <- ifelse(out$meta$treatment == 1, -1, 1)
## low max iteration number just for testing
sts_estimate <- sts(~ treatment*pid_rep, ~ noTreatment, out, K = 3, maxIter = 1, verbose = FALSE)
```

`summary.STS` *Summary Function for the STS objects*

Description

Function to report on the contents of STS objects

Usage

```
## S3 method for class 'STS'
summary(object, ...)
```

Arguments

`object`	An STS object.
`...`	Additional arguments affecting the summary

Details

Summary prints a short statement about the model and then runs `printTopWords`.

`textProcessor` *Process a Vector of Raw Texts*

Description

Function that takes in a vector of raw texts (in a variety of languages) and performs basic operations. This function is essentially a wrapper around the **tm** package where various user specified options can be selected. It is a port of the function of the same name in the stm package, so that a corpus can be prepared for `sts` without installing stm.

Usage

```
textProcessor(
  documents,
  metadata = NULL,
  lowercase = TRUE,
  removestopwords = TRUE,
  removenumbers = TRUE,
  removepunctuation = TRUE,
```

```
  ucp = FALSE,
  stem = TRUE,
  wordLengths = c(3, Inf),
  sparselevel = 1,
  language = "en",
  verbose = TRUE,
  onlycharacter = FALSE,
  striphtml = FALSE,
  customstopwords = NULL,
  custompunctuation = NULL,
  v1 = FALSE
)
```

Arguments

documents	The documents to be processed. A character vector where each entry is the full text of a document (if passed as a different type it will attempt to convert to a character vector).
metadata	Additional data about the documents. Specifically a `data.frame` or `matrix` object with number of rows equal to the number of documents and one column per meta-data type. The column names are used to label the metadata. The metadata do not affect the text processing, but providing the metadata object ensures that if documents are dropped the corresponding metadata rows are dropped as well.
lowercase	Whether all words should be converted to lower case. Defaults to TRUE.
removestopwords	Whether stop words should be removed using the stop word list in **tm**. Defaults to TRUE.
removenumbers	Whether numbers should be removed. Defaults to TRUE.
removepunctuation	whether punctuation should be removed. Defaults to TRUE.
ucp	When TRUE passes `ucp=TRUE` to `tm::removePunctuation` which removes a more general set of punctuation (the Unicode general category P). Defaults to `FALSE`.
stem	Whether or not to stem words. Defaults to TRUE.
wordLengths	From the **tm** package. An integer vector of length 2. Words shorter than the minimum word length `wordLengths[1]` or longer than the maximum word length `wordLengths[2]` are discarded. Defaults to `c(3, Inf)`, i.e., a minimum word length of 3 characters.
sparselevel	Removes terms where at least sparselevel proportion of the entries are 0. Defaults to 1 which effectively turns the feature off.
language	Language used for processing. Defaults to English. `tm` uses the `SnowballC` stemmer which as of version 0.5 supports "danish dutch english finnish french german hungarian italian norwegian portuguese romanian russian spanish swedish turkish". These can be specified as any on of the above strings or by the three-letter ISO-639 codes. You can also set language to "na" if you want to leave it deliberately unspecified (see documentation in `tm`).

`verbose` If true prints information as it processes.

`onlycharacter` When TRUE, runs a regular expression substitution to replace all non-alphanumeric characters. These characters can crash textProcessor for some operating systems. May remove foreign characters depending on encoding. Defaults to FALSE.

`striphtml` When TRUE, runs a regular expression substitution to strip html contained within <>. Defaults to FALSE.

`customstopwords`
A character vector containing words to be removed. Defaults to NULL which does not remove any additional words. This function is primarily for easy removal of application specific stopwords. Note that as with other stopwords these are removed after converting everything to lower case but before removing numbers, punctuation or stemming. Thus words to be removed should be all lower case but otherwise complete.

`custompunctuation`
A character vector containing any characters to be removed immediately after standard punctuation removal. This function exists primarily for easy removal of application specific punctuation not caught by the punctuation filter (although in principle it could be used to remove any term). If you pass a single character string starting with an open bracket, it is treated as a regular expression; otherwise the characters are combined into a bracketed character class.

`v1` A logical which defaults to `FALSE`. If set to `TRUE` it will use the ordering of operations used in Version 1.0 of the stm package to remove punctuation. Primarily for replication purposes.

Details

This function is designed to provide a convenient and quick way to process a relatively small volume of texts for analysis with the package. It is designed to quickly ingest data in a simple form like a spreadsheet where each document sits in a single cell. Once the text has been processed by `textProcessor`, it is usually passed to `prepDocuments`.

The processor always strips extra white space and removes non-printing characters. Other operations (lower-casing, stop word, number and punctuation removal, and stemming) are optional. Stemming uses the Snowball stemmer from the **SnowballC** package via `tm::stemDocument`.

Value

A list containing

`documents` A list of documents in the format used by `sts`.

`vocab` Character vector of vocabulary.

`meta` Data frame or matrix containing the user-supplied metadata for the retained documents.

`docs.removed` Indices of the original documents that were dropped because they contained no words after processing.

References

Ingo Feinerer and Kurt Hornik (2013). tm: Text Mining Package. R package version 0.5-9.1.

Ingo Feinerer, Kurt Hornik, and David Meyer (2008). Text Mining Infrastructure in R. *Journal of Statistical Software* 25(5): 1-54.

Roberts, M. E., Stewart, B. M., and Tingley, D. (2019). "stm: An R Package for Structural Topic Models." *Journal of Statistical Software* 91(2): 1-40.

See Also

prepDocuments

Examples

```
head(gadarian)
temp <- textProcessor(documents = gadarian$open.ended.response,
                      metadata = gadarian, verbose = FALSE)
out <- prepDocuments(temp$documents, temp$vocab, temp$meta)
```

---

topicExclusivity *Compute Exclusivity*

Description

Calculate an exclusivity metric for an STS model.

Usage

```
topicExclusivity(object, M = 10, frexw = 0.7)
```

Arguments

object	Model output from sts
M	the number of top words to consider per topic
frexw	the frex weight

Details

Roberts et al 2014 proposed an exclusivity measure to help with topic model selection.

The exclusivity measure includes some information on word frequency as well. It is based on the FREX labeling metric (see Roberts et al. 2014) with the weight set to .7 in favor of exclusivity by default.

Value

a numeric vector containing exclusivity for each topic

References

Mimno, D., Wallach, H. M., Talley, E., Leenders, M., and McCallum, A. (2011, July). "Optimizing semantic coherence in topic models." In Proceedings of the Conference on Empirical Methods in Natural Language Processing (pp. 262-272). Association for Computational Linguistics. Chicago

Bischof and Airoldi (2012) "Summarizing topical content with word frequency and exclusivity" In Proceedings of the International Conference on Machine Learning.

Roberts, M., Stewart, B., Tingley, D., Lucas, C., Leder-Luis, J., Gadarian, S., Albertson, B., et al. (2014). "Structural topic models for open ended survey responses." American Journal of Political Science, 58(4), 1064-1082.

Examples

```
#An example using the Gadarian data.
# From Raw text to fitted model using textProcessor() which leverages the
# tm Package
library("sts")
temp<-textProcessor(documents=gadarian$open.ended.response,
metadata=gadarian, verbose = FALSE)
out <- prepDocuments(temp$documents, temp$vocab, temp$meta, verbose = FALSE)
out$meta$noTreatment <- ifelse(out$meta$treatment == 1, -1, 1)
## low max iteration number just for testing
sts_estimate <- sts(~ treatment*pid_rep, ~ noTreatment, out, K = 3, maxIter = 2)
topicExclusivity(sts_estimate)
```

`topicSemanticCoherence` *Compute Semantic Coherence*

Description

Calculates semantic coherence for an STS model.

Usage

```
topicSemanticCoherence(object, corpus, M = 10)
```

Arguments

`object`	Model output from sts
`corpus`	The document term matrix to be modeled in a sparse term count matrix with one row per document and one column per term. The object must be a list of with each element corresponding to a document. Each document is represented as an integer matrix with two rows, and columns equal to the number of unique vocabulary words in the document. The first row contains the 1-indexed vocabulary entry and the second row contains the number of times that term appears. This is the format produced by `prepDocuments`.
`M`	the number of top words to consider per topic

Value

a numeric vector containing semantic coherence for each topic

Examples

```
#An example using the Gadarian data.  From raw text to
# fitted model using textProcessor() which leverages the tm Package
library("sts")
temp<-textProcessor(documents=gadarian$open.ended.response,
metadata=gadarian, verbose = FALSE)
out <- prepDocuments(temp$documents, temp$vocab, temp$meta, verbose = FALSE)
out$meta$noTreatment <- ifelse(out$meta$treatment == 1, -1, 1)
## low max iteration number just for testing
sts_estimate <- sts(~ treatment*pid_rep, ~ noTreatment, out, K = 3, maxIter = 2, verbose = FALSE)
topicSemanticCoherence(sts_estimate, out)
```

Index

∗ **datasets**
gadarian, 5
∗ **package**
sts-package, 2

estimateRegns, *3*, 3, *15*

findRepresentativeDocs, *3*, 4

gadarian, 5

heldoutLikelihood, *3*, 6, *7*

make.heldout, *3*, *6*, 7

plot.STS, *3*, 8
plotRepresentativeDocs, *3*, 9
prepDocuments, *3*, 10, *14*, *18–20*
print.STS *(*summary.STS*)*, 16
printRegnTables, 11
printTopWords, *3*, 12, *16*

sts, *3*, *7*, *10, 11*, 13, *16*, *18*
sts-package, 2
summary.STS, 16

textProcessor, *3*, *11*, 16
topicExclusivity, *3*, 19
topicSemanticCoherence, *3*, 20