Package 'sts'

November 6, 2024

Type Package

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

Version 1.1

Date 2024-11-04

Author Shawn Mankad [aut, cre], Li Chen [aut]

Maintainer Shawn Mankad <smankad@ncsu.edu>

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 (2024) <doi:10.1287/mnsc.2022.00261>.

License MIT + file LICENSE

Imports Rcpp, RcppArmadillo, glmnet, matrixStats, slam, foreach, doParallel, parallel, stm, Matrix, mvtnorm, ggplot2

Suggests tm

LinkingTo Rcpp, RcppArmadillo

Encoding UTF-8

RoxygenNote 7.2.1

NeedsCompilation yes

Repository CRAN

Date/Publication 2024-11-06 16:10:02 UTC

Contents

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

Author(s)

Author: Shawn Mankad and Li Chen

Maintainer: Shawn Mankad <smankad@ncsu.edu>

References

Chen L. and Mankad, S. (forthcoming) "A Structural Topic and Sentiment-Discourse Model for Text Analysis" Management Science.

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 same format in the stm package.

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("tm"); library("stm"); 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 plotting 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("tm"); library("stm"); 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)
```

heldoutLikelihood *Heldout Log-Likelihood*

Description

Compute the heldout log-likelihood of the STS model

Usage

```
heldoutLikelihood(mv, kappa, alpha, missing)
```

Arguments

mv	the baseline log-transformed occurrence rate of each word in the corpus
kappa	the estimated kappa coefficients
alpha	the estimated alpha values for the corpus
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("tm"); library("stm"); 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)
## low max iteration number just for testing
sts_estimate <- sts(~ treatment*pid_rep, ~ noTreatment, out, K = 3, maxIter = 2)
sm <- sample(x=1:length(out_ho$missing$index),
size = length(out_ho$missing$index)*0.8, replace = TRUE)
d.h <- list(index = out_ho$missing$index[sm], docs = out_ho$missing$docs[sm])
heldoutLikelihood(mv=sts_estimate$mv, kappa=sts_estimate$kappa,
alpha=sts_estimate$alpha, missing=d.h)$expected.heldout
```

---

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("tm"); library("stm"); 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

Function for plotting 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("tm"); library("stm"); 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)
```

---

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("tm"); library("stm"); 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 *Function for printing 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("tm"); library("stm"); 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 = "adjusted",
  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 same format in 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.
kappaEstimation	A character input specifying how kappa should be estimated. "lasso" allows for penalties on the L1 norm. We estimate a regularization path and then select the optimal shrinkage parameter using AIC. "adjusted" (default) 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 Li and Mankad (forthcoming) 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 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. (forthcoming) "A Structural Topic and Sentiment-Discourse Model for Text Analysis" Management Science.

See Also

estimateRegns

Examples

```
#An example using the Gadarian data from the stm package.  From Raw text to
# fitted model using textProcessor() which leverages the tm Package
library("tm"); library("stm"); 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)
```

---

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.

`topicExclusivity` *Exclusivity*

Description

Calculate an exclusivity metric for an STS model.

Usage

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

Arguments

`beta`	the beta probability matrix (topic-word distributions) for a given document or alpha-level
`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 the stm package.
# From Raw text to fitted model using textProcessor() which leverages the
# tm Package
library("tm"); library("stm"); 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)
full_beta_distn <- exp(sts_estimate$mv + sts_estimate$kappa$kappa_t +
sts_estimate$kappa$kappa_s %*% diag(apply(sts_estimate$alpha[,3:5], 2, mean)))
full_beta_distn <- t(apply(full_beta_distn, 1,
function(m) m / colSums(full_beta_distn)))
topicExclusivity(full_beta_distn)
```

`topicSemanticCoherence` *Semantic Coherence*

Description

Calculates semantic coherence for an STS model.

Usage

```
topicSemanticCoherence(beta, documents, vocab, M = 10)
```

Arguments

`beta`	the beta probability matrix (topic-word distributions) for a given document or alpha-level
`documents`	the documents over which to calculate coherence
`vocab`	the vocabulary corresponding to the terms in the beta matrix
`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 the stm package.  From Raw text to
# fitted model using textProcessor() which leverages the tm Package
library("tm"); library("stm"); 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)
full_beta_distn <- exp(sts_estimate$mv + sts_estimate$kappa$kappa_t +
sts_estimate$kappa$kappa_s %*% diag(apply(sts_estimate$alpha[,3:5], 2, mean)))
full_beta_distn <- t(apply(full_beta_distn, 1,
function(m) m / colSums(full_beta_distn)))
topicSemanticCoherence(full_beta_distn, out$documents, out$vocab)
```

Index