Package 'sts'

September 17, 2024

Type Package

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

Version 1.0

Date 2024-09-04

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 Li and Mankad (2024) <doi:10.2139/ssrn.4020651>.

License MIT + file LICENSE

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

Suggests tm

LinkingTo Rcpp, RcppArmadillo

Encoding UTF-8

RoxygenNote 7.2.1

NeedsCompilation yes

Repository CRAN

Date/Publication 2024-09-17 11:50:01 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: `estimateRegnTables` `exclusivitySTS` `semanticCoherenceSTS` `heldoutLikelihood`

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`

estimateRegnTables *Regression Table Estimation*

Description

Estimates regression tables for prevalence and sentiment/discourse.

Usage

```
estimateRegnTables(alpha, mu, sigma, X)
```

Arguments

alpha	the estimated alpha variable values for each document
mu	the mean (fitted) values for alpha based on document-level variables * estimated Gamma for each document
sigma	the estimated covariance matrix for the alpha parameters
X	the covariates used to estimate the STS model

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)
X <- model.matrix(~1+out$meta$treatment + out$meta$pid_rep +
out$meta$treatment * out$meta$pid_rep)[,-1]
X_seed <- as.matrix(out$meta$treatment)
## low max iteration number just for testing
sts_estimate <- sts(X, X_seed, out, numTopics = 3, verbose = FALSE,
parallelize = FALSE, maxIter = 3, initialization = 'anchor')
regn_tables <- estimateRegnTables(sts_estimate$alpha, mu = sts_estimate$mu,
sigma = sts_estimate$sigma, X = X)
printRegnTables(regn_tables)
```

`exclusivitySTS` *Exclusivity*

Description

Calculate an exclusivity metric for an STS model.

Usage

```
exclusivitySTS(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)
X <- model.matrix(~1+out$meta$treatment + out$meta$pid_rep +
out$meta$treatment * out$meta$pid_rep)[,-1]
X_seed <- as.matrix(out$meta$treatment)
## low max iteration number just for testing
sts_estimate <- sts(X, X_seed, out, numTopics = 3, verbose = FALSE,
parallelize = FALSE, maxIter = 3, initialization = 'anchor')
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)))
exclusivitySTS(full_beta_distn)
```

`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)
X <- model.matrix(~1+out$meta$treatment + out$meta$pid_rep +
out$meta$treatment * out$meta$pid_rep)[,-1]
X_seed <- as.matrix(out$meta$treatment)
out <- make.heldout(out$documents, out$vocab)
## low max iteration number just for testing
sts_estimate <- sts(X, X_seed, out, numTopics = 3, verbose = FALSE,
parallelize = FALSE, maxIter = 3, initialization = 'anchor')
sm <- sample(x=1:length(out$missing$index),
size = length(out$missing$index)*0.8, replace = TRUE)
d.h <- list(index = out$missing$index[sm], docs = out$missing$docs[sm])
heldoutLikelihood(mv=sts_estimate$mv, kappa=sts_estimate$kappa,
alpha=sts_estimate$alpha, missing=d.h)$expected.heldout
```

---

printRegnTables *Print estimated regression tables*

Description

Prints estimated regression tables from estimateRegnTables()

Usage

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

Arguments

x	the estimated regression tables from estimateRegnTables()
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)
X <- model.matrix(~1+out$meta$treatment + out$meta$pid_rep +
out$meta$treatment * out$meta$pid_rep)[,-1]
X_seed <- as.matrix(out$meta$treatment)
## low max iteration number just for testing
sts_estimate <- sts(X, X_seed, out, numTopics = 3, verbose = FALSE,
parallelize = FALSE, maxIter = 3, initialization = 'anchor')
regn_tables <- estimateRegnTables(sts_estimate$alpha, mu = sts_estimate$mu,
sigma = sts_estimate$sigma, X = X)
printRegnTables(regn_tables)
```

---

semanticCoherenceSTS *Semantic Coherence*

Description

Calculates semantic coherence for an STS model.

Usage

```
semanticCoherenceSTS(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)
X <- model.matrix(~1+out$meta$treatment + out$meta$pid_rep +
```

```
out$meta$treatment * out$meta$pid_rep)[,-1]
X_seed <- as.matrix(out$meta$treatment)
## low max iteration number just for testing
sts_estimate <- sts(X, X_seed, out, numTopics = 3, verbose = FALSE,
parallelize = FALSE, maxIter = 3, initialization = 'anchor')
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)))
semanticCoherenceSTS(full_beta_distn, out$documents, out$vocab)
```

`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(
  X,
  X_seed,
  corpus,
  numTopics,
  maxIter = 100,
  initialization = "stm",
  estimation = "lasso",
  verbose = TRUE,
  parallelize = FALSE,
  stmSeed = NULL
)
```

Arguments

`X`	Data frame of document-specific content covariates affect how much (prevalence) and the way in which a topic is discussed (sentiment-discourse).
`X_seed`	A vector of length equal to the corpus size. This is the key experimental variable (e.g., review rating or 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.

numTopics	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.
initialization	Character argument that allows the user to specify an initialization method. The default choice, "stm", uses a fitted STM model (Roberts et al. 2014, 2016) to initialize coefficients related to prevalence and sentiment-discourse. One can also choose "anchor" to initialize prevalence according to anchor words and the key experimental covariate identified in argument X_seed.
estimation	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" does not utilize the lasso penalty. 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

estimateRegnTables

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)
X <- model.matrix(~1+out$meta$treatment + out$meta$pid_rep +
out$meta$treatment * out$meta$pid_rep)[,-1]
X_seed <- as.matrix(out$meta$treatment)
## low max iteration number just for testing
sts_estimate <- sts(X, X_seed, out, numTopics = 3, verbose = FALSE,
parallelize = FALSE, maxIter = 3, initialization = 'anchor')
```

Index