Package 'stm'

August 21, 2023

Title Estimation of the Structural Topic Model

Version 1.3.6.1

Description The Structural Topic Model (STM) allows researchers to estimate topic models with document-level covariates. The package also includes tools for model selection, visualization, and estimation of topic-covariate regressions. Methods developed in Roberts et. al. (2014) <doi:10.1111/ajps.12103> and Roberts et. al. (2016) <doi:10.1080/01621459.2016.1141684>. Vignette is Roberts et. al. (2019) <doi:10.18637/jss.v091.i02>.

Depends R (>= 3.5.0), methods

LinkingTo Rcpp, RcppArmadillo

Imports Rcpp (>= 0.11.3), data.table, glmnet, grDevices, graphics, lda, Matrix, matrixStats, parallel, quadprog, quanteda, slam, splines, stats, stringr, utils

Suggests clue, geometry, huge, igraph, LDAvis, KernSmooth, NLP, rsvd, Rtsne, SnowballC, spelling, testthat, tm (>= 0.6), wordcloud

Encoding UTF-8

LazyData yes

License MIT + file LICENSE

URL http://www.structuraltopicmodel.com/

BugReports https://github.com/bstewart/stm/issues

RoxygenNote 7.1.1

Language en-US

NeedsCompilation yes

Author Margaret Roberts [aut], Brandon Stewart [aut, cre], Dustin Tingley [aut], Kenneth Benoit [ctb]

Maintainer Brandon Stewart <bms4@princeton.edu>

Repository CRAN

Date/Publication 2023-08-21 07:00:41 UTC

R topics documented:

stm-package . . . 3
alignCorpus . . . 4
asSTMCorpus . . . 5
checkBeta . . . 6
checkResiduals . . . 7
cloud . . . 8
convertCorpus . . . 10
estimateEffect . . . 11
findThoughts . . . 13
findTopic . . . 15
fitNewDocuments . . . 16
gadarian . . . 20
labelTopics . . . 21
make.dt . . . 22
make.heldout . . . 23
manyTopics . . . 24
multiSTM . . . 26
optimizeDocument . . . 30
permutationTest . . . 32
plot.estimateEffect . . . 34
plot.MultimodDiagnostic . . . 38
plot.searchK . . . 40
plot.STM . . . 41
plot.STMpermute . . . 44
plot.topicCorr . . . 45
plotModels . . . 46
plotQuote . . . 47
plotRemoved . . . 48
plotTopicLoess . . . 49
poliblog5k . . . 51
prepDocuments . . . 52
readCorpus . . . 54
readLdac . . . 55
s . . . 56
sageLabels . . . 56
searchK . . . 57
selectModel . . . 59
stm . . . 62
summary.estimateEffect . . . 70
summary.STM . . . 71
textProcessor . . . 71
thetaPosterior . . . 74
toLDAvis . . . 76
toLDAvisJson . . . 77
topicCorr . . . 79
topicLasso . . . 80

topicQuality 82
writeLdac 84

Index **85**

`stm-package` *Structural Topic Model*

Description

This package implements the Structural Topic Model, a general approach to including document-level metadata within mixed-membership topic models. To read the vignette use `vignette('stmVignette')`.

Details

Functions to manipulate documents: `textProcessor` `readCorpus` `prepDocuments`

Functions to fit the model: `stm` `selectModel` `manyTopics` `searchK`

Functions to summarize a model: `labelTopics` `summary.STM` `findThoughts`

Functions for Post-Estimation: `estimateEffect` `topicCorr` `permutationTest`

Plotting Functions: `plot.STM` `plot.estimateEffect` `plot.topicCorr` `plot.STMpermute` `plotQuote` `plotTopicLoess` `plotModels` `topicQuality`

Pre-Fit Models and Data: `gadarian` `gadarianFit` `poliblog5k`

Author(s)

Author: Margaret E. Roberts, Brandon M. Stewart and Dustin Tingley

Maintainer: Brandon Stewart <bms4@princeton.edu>

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., Tingley, D., Lucas, C., Leder-Luis, J., Gadarian, S., Albertson, B., Albertson, B. and Rand, D. (2014). "Structural topic models for open ended survey responses." American Journal of Political Science.

Additional papers at: structuraltopicmodel.com

See Also

`stm`

alignCorpus *Align the vocabulary of a new corpus to an old corpus*

Description

Function that takes in a list of documents, vocab and (optionally) metadata for a corpus of previously unseen documents and aligns them to an old vocabulary. Helps preprocess documents for fitNewDocuments.

Usage

```
alignCorpus(new, old.vocab, verbose = TRUE)
```

Arguments

new	a list (such as those produced by textProcessor or prepDocuments) containing a list of documents in stm format, a character vector containing the vocabulary and optional a data.frame containing meta data. These should be labeled documents, vocab,and meta respectively. This is the new set of unseen documents which will be returned with the vocab renumbered and all words not appearing in old removed.
old.vocab	a character vector containing the vocabulary that you want to align to. In general this will be the vocab used in your original stm model fit which from an stm object called mod can be accessed as mod$vocab.
verbose	a logical indicating whether information about the new corpus should be printed to the screen. Defaults to TRUE.

Details

When estimating topic proportions for previously unseen documents using fitNewDocuments the new documents must have the same vocabulary ordered in the same was as the original model. This function helps with that process.

Note: the code is not really built for speed or memory efficiency- if you are trying to do this with a really large corpus of new texts you might consider building the object yourself using **quanteda** or some other option.

Value

documents	A list containing the documents in the stm format.
vocab	Character vector of vocabulary.
meta	Data frame or matrix containing the user-supplied metadata for the retained documents.
docs.removed	document indices (corresponding to the original data passed) of documents removed because they contain no words
words.removed	words dropped from new

tokens.removed	the total number of tokens dropped from the new documents.
wordcounts	counts of times the old vocab appears in the new documents
prop.overlap	length two vector used to populate the message printed by verbose.

See Also

prepDocuments fitNewDocuments

Examples

```
#we process an original set that is just the first 100 documents
temp<-textProcessor(documents=gadarian$open.ended.response[1:100],metadata=gadarian[1:100,])
out <- prepDocuments(temp$documents, temp$vocab, temp$meta)
set.seed(02138)
#Maximum EM its is set low to make this run fast, run models to convergence!
mod.out <- stm(out$documents, out$vocab, 3, prevalence=~treatment + s(pid_rep),
              data=out$meta, max.em.its=5)
#now we process the remaining documents
temp<-textProcessor(documents=gadarian$open.ended.response[101:nrow(gadarian)],
                    metadata=gadarian[101:nrow(gadarian),])
#note we don't run prepCorpus here because we don't want to drop any words- we want
#every word that showed up in the old documents.
newdocs <- alignCorpus(new=temp, old.vocab=mod.out$vocab)
#we get some helpful feedback on what has been retained and lost in the print out.
#and now we can fit our new held-out documents
fitNewDocuments(model=mod.out, documents=newdocs$documents, newData=newdocs$meta,
                origData=out$meta, prevalence=~treatment + s(pid_rep),
                prevalencePrior="Covariate")
```

---

asSTMCorpus *STM Corpus Coercion*

Description

Convert a set of document term counts and associated metadata to the form required for processing by the stm function.

Usage

```
asSTMCorpus(documents, vocab, data = NULL, ...)
```

Arguments

documents	A documents-by-term matrix of counts, or a set of counts in the format returned by prepDocuments. Supported matrix formats include **quanteda** dfm and **Matrix** sparse matrix objects in "dgCMatrix" or "dgTMatrix" format.

vocab	Character vector specifying the words in the corpus in the order of the vocab indices in documents. Each term in the vocabulary index must appear at least once in the documents. See prepDocuments for dropping unused items in the vocabulary. If documents is a sparse matrix or **quanteda** dfm object, then vocab should not (and must not) be supplied. It is contained already inside the column names of the matrix.
data	An optional data frame containing the prevalence and/or content covariates. If unspecified the variables are taken from the active environment.
...	Additional arguments passed to or from other methods.

Value

A list with components "documents", "vocab", and "data" in the form needed for further processing by the stm function.

See Also

prepDocuments, stm

Examples

```
library(quanteda)
gadarian_corpus <- corpus(gadarian, text_field = "open.ended.response")
gadarian_dfm <- dfm(gadarian_corpus,
                    remove = stopwords("english"),
                    stem = TRUE)
asSTMCorpus(gadarian_dfm)
```

---

checkBeta *Looks for words that load exclusively onto a topic*

Description

Checks the log beta matrix for values too close to 0, which reflect words that load onto a single topic.

Usage

```
checkBeta(stmobject, tolerance = 0.01)
```

Arguments

stmobject	STM Model Output
tolerance	User specified input reflecting closeness to 1. E.g. a tolerance of .01 will flag any values greater than .99. Tolerance must be above 1e-6.

Details

The function checks the log beta matrix for values that exceed the tolerance threshold, indicating that a word has loaded onto a single topics. The output gives the user lists of which topics have problems, which words in which topics have problems, as well as a count of the total problems in topics and the total number of problem words.

Note that if the tolerance value is below 1e-6, this function will throw an error.

Value

`problemTopics`	A list of vectors, each vector corresponding to the set of topics in the relevant beta matrix that contain words with too high of a loading to that topic
`topicErrorTotal`	A list of integers, each corresponding to the total number of topics with problems in the relevant beta matrix
`problemWords`	A list of matrices, each corresponding to a relevant beta matrix, which gives the topic and word index of each word with too high of a topic loading
`wordErrorTotal`	A list of integers, each corresponding to the total words with problems for the relevant beta matrix
`check`	A boolean representing if the check was passed. If wordErrorTotal is all 0s (no errors), check is True.

Author(s)

Antonio Coppola

Examples

```
checkBeta(gadarianFit)
```

---

checkResiduals *Residual dispersion test for topic number*

Description

Computes the multinomial dispersion of the STM residuals as in Taddy (2012)

Usage

```
checkResiduals(stmobj, documents, tol = 0.01)
```

Arguments

`stmobj`	An `STM` model object for which to compute residuals.
`documents`	The documents corresponding to `stmobj` as in `stm`.
`tol`	The tolerance parameter for calculating the degrees of freedom. Defaults to 1/100 as in Taddy(2012)

Details

This function implements the residual-based diagnostic method of Taddy (2012). The basic idea is that when the model is correctly specified the multinomial likelihood implies a dispersion of the residuals: $\sigma^2 = 1$. If we calculate the sample dispersion and the value is greater than one, this implies that the number of topics is set too low, because the latent topics are not able to account for the overdispersion. In practice this can be a very demanding criterion, especially if the documents are long. However, when coupled with other tools it can provide a valuable perspective on model fit. The function is based on the Taddy 2012 paper as well as code found in maptpx package.

Further details are available in the referenced paper, but broadly speaking the dispersion is derived from the mean of the squared adjusted residuals. We get the sample dispersion by dividing by the degrees of freedom parameter. In estimating the degrees of freedom, we follow Taddy (2012) in approximating the parameter $\hat{N}$ by the number of expected counts exceeding a tolerance parameter. The default value of 1/100 given in the Taddy paper can be changed by setting the `tol` argument.

The function returns the estimated sample dispersion (which equals 1 under the data generating process) and the p-value of a chi-squared test where the null hypothesis is that $\sigma^2 = 1$ vs the alternative $\sigma^2 > 1$. As Taddy notes and we echo, rejection of the null 'provides a very rough measure for evidence in favor of a larger number of topics.'

References

Taddy, M. 'On Estimation and Selection for Topic Models'. AISTATS 2012, JMLR W&CP 22

Examples

```
#An example using the Gadarian data.  From Raw text to fitted model.
temp<-textProcessor(documents=gadarian$open.ended.response,metadata=gadarian)
meta<-temp$meta
vocab<-temp$vocab
docs<-temp$documents
out <- prepDocuments(docs, vocab, meta)
docs<-out$documents
vocab<-out$vocab
meta <-out$meta
set.seed(02138)
#maximum EM iterations set very low so example will run quickly.
#Run your models to convergence!
mod.out <- stm(docs, vocab, 3, prevalence=~treatment + s(pid_rep), data=meta,
               max.em.its=5)
checkResiduals(mod.out, docs)
```

`cloud` *Plot a wordcloud*

Description

Use the **wordcloud** package to plot a wordcloud for a particular topic

Usage

```
cloud(
  stmobj,
  topic = NULL,
  type = c("model", "documents"),
  documents,
  thresh = 0.9,
  max.words = 100,
  ...
)
```

Arguments

stmobj	The STM model object to be used in making the word cloud.
topic	NULL to plot the marginal distribution of words in the corpus, or a single integer indicating the topic number.
type	Specifies how the wordcloud is constructed. The type "model" which is used by default is based on the probability of the word given the topic. The type "documents" plots words within documents that have a topic proportion of higher than thresh. This requires that the documents argument also be specified.
documents	The documents object of the same kind as passed to stm. This is only necessary if type="documents".
thresh	The threshold for including a document in the type="documents" setting.
max.words	The maximum number of words to be plotted.
...	Additional parameters passed to wordcloud.

Details

Uses the **wordcloud** package to make a word cloud of a particular topic. The option "model" uses the topic-word model parameters. Thus it shows words weighted by their probability conditional that the word comes from a particular topic. With content covariates it averages over the values for all levels of the content covariate weighted by the empirical frequency in the dataset. The option "documents" plots the words which appear in documents that have a topic proportion higher than thresh. Thus "model" gives a pure model based interpretation of the topic while "documents" gives a picture of all the words in documents which are highly associated with the topic.

References

Ian Fellows (2014). wordcloud: Word Clouds. R package version 2.5. https://cran.r-project.org/package=wordcloud

See Also

plot.STM

Examples

```
cloud(gadarianFit, 1)
```

---

convertCorpus *Convert* **stm** *formatted documents to another format*

Description

Takes an **stm** formatted documents and vocab object and returns formats usable in other packages.

Usage

```
convertCorpus(documents, vocab, type = c("slam", "lda", "Matrix"))
```

Arguments

documents	the documents object in **stm** format
vocab	the vocab object in **stm** format
type	the output type desired. See Details.

Details

We also recommend the **quanteda** and **tm** packages for text preparation etc. The convertCorpus function is provided as a helpful utility for moving formats around, but if you intend to do text processing with a variety of output formats, you likely want to start with **quanteda** or **tm**.

The various type conversions are described below:

type = "slam" Converts to the simple triplet matrix representation used by the **slam** package. This is the format used internally by **tm**.

type = "lda" Converts to the format used by the **lda** package. This is a very minor change as the format in **stm** is based on **lda**'s data representation. The difference as noted in stm involves how the numbers are indexed. Accordingly this type returns a list containing the new documents object and the unchanged vocab object.

type = "Matrix" Converts to the sparse matrix representation used by **Matrix**. This is the format used internally by numerous other text analysis packages.

If you want to write out a file containing the sparse matrix representation popularized by David Blei's C code ldac see the function writeLdac.

See Also

writeLdac readCorpus poliblog5k

Examples

```
#convert the poliblog5k data to slam package format
poliSlam <- convertCorpus(poliblog5k.docs, poliblog5k.voc, type="slam")
class(poliSlam)
poliMatrix <- convertCorpus(poliblog5k.docs, poliblog5k.voc, type="Matrix")
class(poliMatrix)
poliLDA <- convertCorpus(poliblog5k.docs, poliblog5k.voc, type="lda")
str(poliLDA)
```

`estimateEffect` *Estimates regressions using an STM object*

Description

Estimates a regression where documents are the units, the outcome is the proportion of each document about a topic in an STM model and the covariates are document-meta data. This procedure incorporates measurement uncertainty from the STM model using the method of composition.

Usage

```
estimateEffect(
  formula,
  stmobj,
  metadata = NULL,
  uncertainty = c("Global", "Local", "None"),
  documents = NULL,
  nsims = 25,
  prior = NULL
)
```

Arguments

`formula`	A formula for the regression. It should have an integer or vector of numbers on the left-hand side and an equation with covariates on the right hand side. See Details for more information.
`stmobj`	Model output from STM
`metadata`	A dataframe where all predictor variables in the formula can be found. If `NULL` R will look for the variables in the global namespace. It will not look for them in the `STM` object which for memory efficiency only stores the transformed design matrix and thus will not in general have the original covariates.
`uncertainty`	Which procedure should be used to approximate the measurement uncertainty in the topic proportions. See details for more information. Defaults to the Global approximation.
`documents`	If uncertainty is set to `Local`, the user needs to provide the documents object (see `stm` for format).

`nsims` The number of simulated draws from the variational posterior. Defaults to 25. This can often go even lower without affecting the results too dramatically.

`prior` This argument allows the user to specify a ridge penalty to be added to the least squares solution for the purposes of numerical stability. If its a scalar it is added to all coefficients. If its a matrix it should be the prior precision matrix (a diagonal matrix of the same dimension as the `ncol(X)`). When the design matrix is collinear but this argument is not specified, a warning will pop up and the function will estimate with a small default penalty.

Details

This function performs a regression where topic-proportions are the outcome variable. This allows us to conditional expectation of topic prevalence given document characteristics. Use of the method of composition allows us to incorporate our estimation uncertainty in the dependent variable. Mechanically this means we draw a set of topic proportions from the variational posterior, compute our coefficients, then repeat. To compute quantities of interest we simulate within each batch of coefficients and then average over all our results.

The formula specifies the nature of the linear model. On the left hand-side we use a vector of integers to indicate the topics to be included as outcome variables. If left blank then the default of all topics is used. On the right hand-side we can specify a linear model of covariates including standard transformations. Thus the model `2:4 ~ var1 + s(var2)` would indicate that we want to run three regressions on Topics 2, 3 and 4 with predictor variables `var1` and a b-spline transformed `var2`. We encourage the use of spline functions for non-linear transformations of variables.

The function allows the user to specify any variables in the model. However, we caution that for the assumptions of the method of composition to be the most plausible the topic model should contain at least all the covariates contained in the `estimateEffect` regression. However the inverse need not be true. The function will automatically check whether the covariate matrix is singular which generally results from linearly dependent columns. Some common causes include a factor variable with an unobserved level, a spline with degrees of freedom that are too high, or a spline with a continuous variable where a gap in the support of the variable results in several empty basis functions. In these cases the function will still estimate by adding a small ridge penalty to the likelihood. However, we emphasize that while this will produce an estimate it is only identified by the penalty. In many cases this will be an indication that the user should specify a different model.

The function can handle factors and numeric variables. Dates should be converted to numeric variables before analysis.

We offer several different methods of incorporating uncertainty. Ideally we would want to use the covariance matrix that governs the variational posterior for each document (ν). The updates for the global parameters rely only on the sum of these matrices and so we do not store copies for each individual document. The default uncertainty method `Global` uses an approximation to the average covariance matrix formed using the global parameters. The uncertainty method `Local` steps through each document and updates the parameters calculating and then saving the local covariance matrix. The option `None` simply uses the map estimates for θ and does not incorporate any uncertainty. We strongly recommend the `Global` approximation as it provides the best tradeoff of accuracy and computational tractability.

Effects are plotted based on the results of `estimateEffect` which contains information on how the estimates are constructed. Note that in some circumstances the expected value of a topic proportion given a covariate level can be above 1 or below 0. This is because we use a Normal distribution

rather than something constrained to the range between 0 and 1. If a continuous variable goes above 0 or 1 within the range of the data it may indicate that a more flexible non-linear specification is needed (such as using a spline or a spline with greater degrees of freedom).

Value

parameters	A list of K elements each corresponding to a topic. Each element is itself a list of n elements one per simulation. Each simulation contains the MLE of the parameter vector and the variance covariance matrix
topics	The topic vector
call	The original call
uncertainty	The user choice of uncertainty measure
formula	The formula object
data	The original user provided meta data.
modelframe	The model frame created from the formula and data
varlist	A variable list useful for mapping terms with columns in the design matrix

See Also

plot.estimateEffect summary.estimateEffect

Examples

```
#Just one topic (note we need c() to indicate it is a vector)
prep <- estimateEffect(c(1) ~ treatment, gadarianFit, gadarian)
summary(prep)
plot(prep, "treatment", model=gadarianFit, method="pointestimate")

#three topics at once
prep <- estimateEffect(1:3 ~ treatment, gadarianFit, gadarian)
summary(prep)
plot(prep, "treatment", model=gadarianFit, method="pointestimate")

#with interactions
prep <- estimateEffect(1 ~ treatment*s(pid_rep), gadarianFit, gadarian)
summary(prep)
```

---

findThoughts *Find Thoughts*

Description

Outputs most representative documents for a particular topic. Use this in order to get a better sense of the content of actual documents with a high topical content.

Usage

```
findThoughts(
  model,
  texts = NULL,
  topics = NULL,
  n = 3,
  thresh = NULL,
  where = NULL,
  meta = NULL
)
```

Arguments

model	Model object created by stm.
texts	A character vector where each entry contains the text of a document. Must be in the same order as the documents object. NOTE: This is not the documents which are passed to stm and come out of prepDocuments, this is the actual text of the document.
topics	The topic number or vector of topic numbers for which you want to find thoughts. Defaults to all topics.
n	The number of desired documents to be displayed per topic.
thresh	Sets a minimum threshold for the estimated topic proportion for displayed documents. It defaults to imposing no restrictions.
where	An expression in the form of a data.table query. This is passed to the i argument in data.table and a custom query is passed to j. This cannot be used with thresh. See below for more details.
meta	The meta data object to be used with where.

Details

Returns the top n documents ranked by the MAP estimate of the topic's theta value (which captures the modal estimate of the proportion of word tokens assigned to the topic under the model). Setting the thresh argument allows the user to specify a minimal value of theta for returned documents. Returns document indices and top thoughts.

Sometimes you may want to find thoughts which have more conditions than simply a minimum threshold. For example, you may want to grab all documents which satisfy certain conditions on the metadata or other topics. You can supply a query in the style of **data.table** to the where argument. Note that in data.table variables are referenced by their names in the data.table object. The topics themselves are labeled Topic1, Topic2 etc. If you supply the metadata to the meta argument, you can also query based on any available metadata. See below for examples.

If you want to pass even more complicated queries, you can use the function make.dt to generate a data.table object where you can write your own queries.

The plot.findThoughts function is a shortcut for the plotQuote function.

Value

A `findThoughts` object:

`index`	List with one entry per topic. Each entry is a vector of document indices.
`docs`	List with one entry per topic. Each entry is a character vector of the corresponding texts.

See Also

`plotQuote`

Examples

```
findThoughts(gadarianFit, texts=gadarian$open.ended.response, topics=c(1,2), n=3)

#We can plot findThoughts objects using plot() or plotQuote
thought <- findThoughts(gadarianFit, texts=gadarian$open.ended.response, topics=1, n=3)

#plotQuote takes a set of sentences
plotQuote(thought$docs[[1]])

#we can use the generic plot as a shorthand which will make one plot per topic
plot(thought)

#we can select a subset of examples as well using either approach
plot(thought,2:3)
plotQuote(thought$docs[[1]][2:3])


#gather thoughts for only treated documents
thought <- findThoughts(gadarianFit, texts=gadarian$open.ended.response, topics=c(1,2), n=3,
                        where = treatment==1, meta=gadarian)
plot(thought)
#you can also query in terms of other topics
thought <- findThoughts(gadarianFit, texts=gadarian$open.ended.response, topics=c(1), n=3,
                         where = treatment==1 & Topic2>.2, meta=gadarian)
plot(thought)
#these queries can be really complex if you like
thought <- findThoughts(gadarianFit, texts=gadarian$open.ended.response, topics=c(1), n=3,
                        where = (treatment==1 | pid_rep > .5) & Topic3>.2, meta=gadarian)
plot(thought)
```

`findTopic` *Find topics that contain user specified words.*

Description

Find topics that contain user specified words.

Usage

```
findTopic(
  x,
  list,
  n = 20,
  type = c("prob", "frex", "lift", "score"),
  verbose = TRUE
)
```

Arguments

x	The STM model object to be searched. May also be the output from sageLabels.
list	Character vector containing words to be searched.
n	Number of words to consider
type	Type of words to be searched.
verbose	A logical indicating whether details should be printed to the screen.

See Also

findThoughts

Examples

```
lab <- sageLabels(gadarianFit, n=5)
findTopic(lab, c("poor", "immigr", "peopl"))
findTopic(gadarianFit, c("poor", "immigr", "peopl"))
```

---

fitNewDocuments *Fit New Documents*

Description

A function for predicting thetas for an unseen document based on the previously fit model.

Usage

```
fitNewDocuments(
  model = NULL,
  documents = NULL,
  newData = NULL,
  origData = NULL,
  prevalence = NULL,
  betaIndex = NULL,
  prevalencePrior = c("Average", "Covariate", "None"),
```

```
  contentPrior = c("Average", "Covariate"),
  returnPosterior = FALSE,
  returnPriors = FALSE,
  designMatrix = NULL,
  test = TRUE,
  verbose = TRUE
)
```

Arguments

model	the originally fit STM object.
documents	the new documents to be fit. These documents must be in the stm format and be numbered in the same way as the documents in the original model with the same dimension of vocabulary. See alignCorpus or the **quanteda** feature dfm_select for a way to do this.
newData	the metadata for the prevalence prior which goes with the unseen documents. As in the original data this cannot have any missing data.
origData	the original metadata used to fit the STM object.
prevalence	the original formula passed to prevalence when stm was called. The function will try to reconstruct this.
betaIndex	a vector which indicates which level of the content covariate is used for each unseen document. If originally passed as a factor, this can be passed as a factor or character vector as well but it must not have any levels not included in the original factor.
prevalencePrior	three options described in detail below. Defaults to "Average" when data is not provided and "Covariate" when it is.
contentPrior	two options described in detail below. Defaults to "Average" when betaIndex is not provided and "Covariate" when it is.
returnPosterior	the function always returns the posterior mode of theta (document-topic proportions), If set to TRUE this will return the full variational posterior. Note that this will return a dense K-by-K matrix for every document which can be very memory intensive if you are processing a lot of documents.
returnPriors	the function always returns the options that were set for the prior (either by the user or chosen internally by the defaults). In the case of content covariates using the covariate prior this will be a set of indices to the original beta matrix so as not to make the object too large.
designMatrix	an option for advanced users to pass an already constructed design matrix for prevalence covariates. This will override the options in newData, origData and test. See details below- please do not attempt to use without reading carefully.
test	a test of the functions ability to reconstruct the original functions.
verbose	Should a dot be printed every time 1 percent of the documents are fit.

Details

Due to the existence of the metadata in the model, this isn't as simple as in models without side information such as Latent Dirichlet Allocation. There are four scenarios: models without covariate information, models with prevalence covariates only, models with content covariates only and models with both. When there is not covariate information the choice is essentially whether or not to use prior information.

We offer three types of choices (and may offer more in the future):

"None" No prior is used. In the prevalence case this means that the model simply maximizes the likelihood of seeing the words given the word-topic distribution. This will in general produce more sharply peaked topic distributions than the prior. This can be used even without the covariates. This is not an option for topical content covariate models. If you do not observe the topical content covariate, use the "Average" option.

"Average" We use a prior that is based on the average over the documents in the training set. This does not require the unseen documents to observe the covariates. In a model that originally had covariates we need to adjust our estimate of the variance-covariance matrix sigma to accommodate that we no longer have the covariate information. So we recalculate the variance based on what it would have been if we didn't have any covariates. This helps avoid an edge case where the covariates are extremely influential and we don't want that strength applied to the new covariate-less setting. In the case of content covariates this essentially use the `sageLabels` approach to create a marginalized distribution over words for each topic.

"Covariate" We use the same covariate driven prior that existed in the original model. This requires that the test covariates be observed for all previously unseen documents.

If you fit a document that was used during training with the options to replicate the initial `stm` model fit you will not necessarily get exactly the same result. `stm` updates the topic-word distributions last so they may shifted since the document-topic proportions were updated. If the original model converged, they should be very close.

By default the function returns only the MAP estimate of the normalized document-topic proportions theta. By selecting `returnPrior=TRUE` you can get the various model parameters used to complete the fit. By selecting `returnPosterior=TRUE` you can get the full variational posterior. Please note that the full variational posterior is very memory intensive. For a sense of scale it requires an extra $K^2 + K \times (V' + 1) + 1$ doubles per document where V' is the number of unique tokens in the document.

Testing: Getting the prevalence covariates right in the unseen documents can be tricky. However as long as you leave `test` set to `TRUE` the code will automatically run a test to make sure that everything lines up. See the internal function `makeDesignMatrix` for more on what is going on here.

Passing a Design Matrix Advanced users may wish to circumvent this process and pass their own design matrix possibly because they used their own function for transforming the original input variables. This can be done by passing the design matrix using the `designMatrix` argument The columns need to match the ordering of the design matrix for the original `stm` object. The design matrix in an stm model called `stmobj` can be found in `stmobj$settings$covariates$X` which can in turn be used to check that you have formatted your result correctly. If you are going to try this we recommend that you read the documentation for `makeDesignMatrix` to understand some of the challenges involved.

If you want even more fine-grained control we recommend you directly use the optimization function `optimizeDocument`

Value

an object of class fitNewDocuments

theta	a matrix with one row per document contain the document-topic proportions at the posterior mode
eta	the mean of the variational posterior, only provided when posterior is requested. Matrix of same dimension as theta
nu	a list with one element per document containing the covariance matrix of the variational posterior. Only provided when posterior is requested.
phis	a list with one element per K by V' matrix containing the variational distribution for each token (where V' is the number of unique words in the given document. They are in the order of appearance in the document. For words repeated more than once the sum of the column is the number of times that token appeared. This is only provided if the posterior is requested.
bound	a vector with one element per document containing the approximate variational lower bound. This is only provided if the posterior is requested.
beta	A list where each element contains the unlogged topic-word distribution for each level of the content covariate. This is only provided if prior is requested.
betaindex	a vector with one element per document indicating which element of the beta list the documents pairs with. This is only provided if prior is requested.
mu	a matrix where each column includes the K-1 dimension prior mean for each document. This is only provided if prior is requested.
sigma	a K-1 by K-1 matrix containing the prior covariance. This is only provided if prior is requested.

See Also

alignCorpus optimizeDocument make.heldout makeDesignMatrix

Examples

```
#An example using the Gadarian data.  From Raw text to fitted model.
#(for a case where documents are all not processed at once see the help
# file for alignCorpus)
temp<-textProcessor(documents=gadarian$open.ended.response,metadata=gadarian)
out <- prepDocuments(temp$documents, temp$vocab, temp$meta)
set.seed(02138)
#Maximum EM its is set low to make this run fast, run models to convergence!
mod.out <- stm(out$documents, out$vocab, 3, prevalence=~treatment + s(pid_rep),
               data=out$meta, max.em.its=5)
fitNewDocuments(model=mod.out, documents=out$documents[1:5], newData=out$meta[1:5,],
                origData=out$meta, prevalence=~treatment + s(pid_rep),
                prevalencePrior="Covariate")
```

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.

Format

A data frame with 351 observations on the following 3 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)
#Process the data for analysis.
temp<-textProcessor(documents=gadarian$open.ended.response,metadata=gadarian)
meta<-temp$meta
vocab<-temp$vocab
docs<-temp$documents
out <- prepDocuments(docs, vocab, meta)
docs<-out$documents
vocab<-out$vocab
meta <-out$meta
```

`labelTopics` *Label topics*

Description

Generate a set of words describing each topic from a fitted STM object. Uses a variety of labeling algorithms (see details).

Usage

```
labelTopics(model, topics = NULL, n = 7, frexweight = 0.5)
```

Arguments

`model`	An `STM` model object.
`topics`	A vector of numbers indicating the topics to include. Default is all topics.
`n`	The desired number of words (per type) used to label each topic. Must be 1 or greater.
`frexweight`	A weight used in our approximate FREX scoring algorithm (see details).

Details

Four different types of word weightings are printed with label topics.

Highest Prob: are the words within each topic with the highest probability (inferred directly from topic-word distribution parameter β).

FREX: are the words that are both frequent and exclusive, identifying words that distinguish topics. This is calculated by taking the harmonic mean of rank by probability within the topic (frequency) and rank by distribution of topic given word $p(z|w = v)$ (exclusivity). In estimating exclusivity we use a James-Stein type shrinkage estimator of the distribution $p(z|w = v)$. More information can be found in the documentation for the internal function `calcfrex` and `js.estimate`.

Score and Lift are measures provided in two other popular text mining packages. For more information on type Score, see the R package `lda` or the internal function `calcscore`. For more information on type Lift, see the R package `maptpx` or or the internal function `calclift`.

Value

A labelTopics object (list)

`prob`	matrix of highest probability words
`frex`	matrix of highest ranking frex words
`lift`	matrix of highest scoring words by lift
`score`	matrix of best words by score
`topicnums`	a vector of topic numbers which correspond to the rows

See Also

stm plot.STM calcfrex js.estimate calcscore calclift

Examples

```
labelTopics(gadarianFit)
```

---

make.dt *Make a* data.table *of topic proportions.*

Description

Combines the document-topic loadings (theta) with metadata to create a data.table object for easy querying.

Usage

```
make.dt(model, meta = NULL)
```

Arguments

model	The stm model.
meta	Optionally, the metadata object passed to the stm model.

Details

This is a simple utility function that creates a **data.table** object which you can use to create more complicated queries than via findThoughts. Topics are named via the convention Topic#, for example Topic1, Topic2 etc. The object also contains docnum which gives the index of the document so you can set keys without worrying about the texts getting disconnected.

We expect that for the vast majority of users the functionality in findThoughts will be sufficient.

See Also

findThoughts

Examples

```
dt <- make.dt(gadarianFit, meta=gadarian)
#now we can do any query.  For example the 5 least associated documents with Topic 2 in
#the treated group
dt[treatment==0, docnum[order(Topic2, decreasing=FALSE)][1:5]]
```

make.heldout *Heldout Likelihood by Document Completion*

Description

Tools for making and evaluating heldout datasets.

Usage

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

Arguments

documents	the documents to be modeled (see stm 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.

Examples

```
prep <- prepDocuments(poliblog5k.docs, poliblog5k.voc,
                      poliblog5k.meta,subsample=500,
                      lower.thresh=20,upper.thresh=200)
heldout <- make.heldout(prep$documents, prep$vocab)
documents <- heldout$documents
vocab <- heldout$vocab
meta <- prep$meta

stm1<- stm(documents, vocab, 5,
           prevalence =~ rating+ s(day),
           init.type="Random",
           data=meta, max.em.its=5)
```

```
eval.heldout(stm1, heldout$missing)
```

`manyTopics` *Performs model selection across separate STM's that each assume different numbers of topics.*

Description

Works the same as selectModel, except user specifies a range of numbers of topics that they want the model fitted for. For example, models with 5, 10, and 15 topics. Then, for each number of topics, selectModel is run multiple times. The output is then processed through a function that takes a pareto dominant run of the model in terms of exclusivity and semantic coherence. If multiple runs are candidates (i.e., none weakly dominates the others), a single model run is randomly chosen from the set of undominated runs.

Usage

```
manyTopics(
  documents,
  vocab,
  K,
  prevalence = NULL,
  content = NULL,
  data = NULL,
  max.em.its = 100,
  verbose = TRUE,
  init.type = "LDA",
  emtol = 1e-05,
  seed = NULL,
  runs = 50,
  frexw = 0.7,
  net.max.em.its = 2,
  netverbose = FALSE,
  M = 10,
  ...
)
```

Arguments

`documents`	The documents to be modeled. 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 similar to the format in the **lda** package except that (following R convention) the vocabulary is indexed from one. Corpora can be imported using the reader function and manipulated using the `prepDocuments`.

vocab	Character vector specifying the words in the corpus in the order of the vocab indices in documents. Each term in the vocabulary index must appear at least once in the documents. See prepDocuments for dropping unused items in the vocabulary.
K	A vector of positive integers representing the desired number of topics for separate runs of selectModel.
prevalence	A formula object with no response variable or a matrix containing topic prevalence covariates. Use s(), ns() or bs() to specify smooth terms. See details for more information.
content	A formula containing a single variable, a factor variable or something which can be coerced to a factor indicating the category of the content variable for each document.
data	Dataset which contains prevalence and content covariates.
max.em.its	The maximum number of EM iterations. If convergence has not been met at this point, a message will be printed.
verbose	A logical flag indicating whether information should be printed to the screen.
init.type	The method of initialization. See stm.
emtol	Convergence tolerance.
seed	Seed for the random number generator. stm saves the seed it uses on every run so that any result can be exactly reproduced. When attempting to reproduce a result with that seed, it should be specified here.
runs	Total number of STM runs used in the cast net stage. Approximately 15 percent of these runs will be used for running a STM until convergence.
frexw	Weight used to calculate exclusivity
net.max.em.its	Maximum EM iterations used when casting the net
netverbose	Whether verbose should be used when calculating net models.
M	Number of words used to calculate semantic coherence and exclusivity. Defaults to 10.
...	Additional options described in details of stm.

Details

Does not work with models that have a content variable (at this point).

Value

out	List of model outputs the user has to choose from. Take the same form as the output from a stm model.
semcoh	Semantic coherence values for each topic within each model selected for each number of topics.
exclusivity	Exclusivity values for each topic within each model selected. Only calculated for models without a content covariate.

Examples

```
## Not run:

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

set.seed(02138)
storage<-manyTopics(docs,vocab,K=3:4, prevalence=~treatment + s(pid_rep),data=meta, runs=10)
#This chooses the output, a single run of STM that was selected,
#from the runs of the 3 topic model
t<-storage$out[[1]]
#This chooses the output, a single run of STM that was selected,
#from the runs of the 4 topic model
t<-storage$out[[2]]
#Please note that the way to extract a result for manyTopics is different from selectModel.

## End(Not run)
```

`multiSTM` *Analyze Stability of Local STM Mode*

Description

This function performs a suite of tests aimed at assessing the global behavior of an STM model, which may have multiple modes. The function takes in a collection of differently initialized STM fitted objects and selects a reference model against which all others are benchmarked for stability. The function returns an output of S3 class 'MultimodDiagnostic', with associated plotting methods for quick inspection of the test results.

Usage

```
multiSTM(
  mod.out = NULL,
  ref.model = NULL,
  align.global = FALSE,
  mass.threshold = 1,
  reg.formula = NULL,
  metadata = NULL,
  reg.nsims = 100,
  reg.parameter.index = 2,
  verbose = TRUE,
  from.disk = FALSE
)
```

Arguments

mod.out	The output of a selectModel() run. This is a list of model outputs the user has to choose from, which all take the same form as the output from a STM model. Currently only works with models without content covariates.
ref.model	An integer referencing the element of the list in mod.out which contains the desired reference model. When set to the default value of NULL this chooses the model with the largest value of the approximate variational bound.
align.global	A boolean parameter specifying how to align the topics of two different STM fitted models. The alignment is performed by solving the linear sum assignment problem using the Hungarian algorithm. If align.global is set to TRUE, the Hungarian algorithm is run globally on the topic-word matrices of the two models that are being compared. The rows of the matrices are aligned such as to minimize the sum of their inner products. This results in each topic in the current runout being matched to a unique topic in the reference model. If align.global is, conversely, set to FALSE, the alignment problem is solved locally. Each topic in the current runout is matched to the one topic in the reference models that yields minimum inner product. This means that multiple topics in the current runout can be matched to a single topic in the reference model, and does not guarantee that all the topics in the reference model will be matched.
mass.threshold	A parameter specifying the portion of the probability mass of topics to be used for model analysis. The tail of the probability mass is disregarded accordingly. If mass.threshold is different from 1, both the full-mass and partial-mass analyses are carried out.
reg.formula	A formula for estimating a regression for each model in the ensemble, where the documents are the units, the outcome is the proportion of each document about a topic in an STM model, and the covariates are the document-level metadata. The formula should have an integer or a vector of numbers on the left-hand side, and an equation with covariates on the right-hand side. If the left-hand side is left blank, the regression is performed on all topics in the model. The formula is exclusively used for building calls to estimateEffect(), so see the documentation for estimateEffect() for greater detail about the regression procedure. If reg.formula is null, the covariate effect stability analysis routines are not performed. The regressions incorporate uncertainty by using an approximation to the average covariance matrix formed using the global parameters.
metadata	A dataframe where the predictor variables in reg.formula can be found. It is necessary to include this argument if reg.formula is specified.
reg.nsims	The number of simulated draws from the variational posterior for each call of estimateEffect(). Defaults to 100.
reg.parameter.index	If reg.formula is specified, the function analyzes the stability across runs of the regression coefficient for one particular predictor variable. This argument specifies which predictor variable is to be analyzed. A value of 1 corresponds to the intercept, a value of 2 correspond to the first predictor variable in reg.formula, and so on. Support for multiple concurrent covariate effect stability analyses is forthcoming.
verbose	If set to TRUE, the function will report progress.

from.disk — If set to TRUE, multiSTM() will load the input models from disk rather than from RAM. This option is particularly useful for dealing with large numbers of models, and is intended to be used in conjunction with the to.disk option of selectModel(). multiSTM() inspects the current directory for RData files.

Details

The purpose of this function is to automate and generalize the stability analysis routines for topic models that are introduced in Roberts, Margaret E., Brandon M. Stewart, and Dustin Tingley: "Navigating the Local Modes of Big Data: The Case of Topic Models" (2014). For more detailed discussion regarding the background and motivation for multimodality analysis, please refer to the original article. See also the documentation for plot.MultimodDiagnostic for help with the plotting methods associated with this function.

Value

An object of 'MultimodDiagnostic' S3 class, consisting of a list with the following components:

N — The number of fitted models in the list of model outputs that was supplied to the function for the purpose of stability analysis.

K — The number of topics in the models.

glob.max — The index of the reference model in the list of model outputs (mod.out) that was supplied to the function. The reference model is selected as the one with the maximum bound value at convergence.

lb — A list of the maximum bound value at convergence for each of the fitted models in the list of model outputs. The list has length N.

lmat — A K-by-N matrix reporting the L1-distance of each topic from the corresponding one in the reference model. This is defined as:

$$L_1 = \sum_v |\beta_{k,v}^{ref} - \beta_{k,v}^{cand}|$$

Where the beta matrices are the topic-word matrices for the reference and the candidate model.

tmat — A K-by-N matrix reporting the number of "top documents" shared by the reference model and the candidate model. The "top documents" for a given topic are defined as the 10 documents in the reference corpus with highest topical frequency.

wmat — A K-by-N matrix reporting the number of "top words" shared by the reference model and the candidate model. The "top words" for a given topic are defined as the 10 highest-frequency words.

lmod — A vector of length N consisting of the row sums of the lmat matrix.

tmod — A vector of length N consisting of the row sums of the tmat matrix.

wmod — A vector of length N consisting of the row sums of the wmat matrix.

semcoh — Semantic coherence values for each topic within each model in the list of model outputs.

L1mat	A K-by-N matrix reporting the limited-mass L1-distance of each topic from the corresponding one in the reference model. Similar to lmat, but computed using only the top portion of the probability mass for each topic, as specified by the mass.threshol parameter. NULL if mass.treshold==1.
L1mod	A vector of length N consisting of the row means of the L1mat matrix.
mass.threshold	The mass threshold argument that was supplied to the function.
cov.effects	A list of length N containing the output of the run of estimateEffect() on each candidate model with the given regression formula. NULL if no regression formula is given.
var.matrix	A K-by-N matrix containing the estimated variance for each of the fitted regression parameters. NULL if no regression formula is given.
confidence.ratings	A vector of length N, where each entry specifies the proportion of regression coefficient estimates in a candidate model that fall within the .95 confidence interval for the corresponding estimate in the reference model.
align.global	The alignment control argument that was supplied to the function.
reg.formula	The regression formula that was supplied to the function.
reg.nsims	The reg.nsims argument that was supplied to the function.
reg.parameter.index	The reg.parameter.index argument that was supplied to the function.

Author(s)

Antonio Coppola (Harvard University), Brandon Stewart (Princeton University), Dustin Tingley (Harvard University)

References

Roberts, M., Stewart, B., & Tingley, D. (2016). "Navigating the Local Modes of Big Data: The Case of Topic Models. In Data Analytics in Social Science, Government, and Industry." New York: Cambridge University Press.

See Also

plot.MultimodDiagnostic selectModel estimateEffect

Examples

```
## Not run:

# Example using Gadarian data
temp<-textProcessor(documents=gadarian$open.ended.response,
                    metadata=gadarian)
meta<-temp$meta
vocab<-temp$vocab
docs<-temp$documents
out <- prepDocuments(docs, vocab, meta)
```

```
docs<-out$documents
vocab<-out$vocab
meta <-out$meta
set.seed(02138)
mod.out <- selectModel(docs, vocab, K=3,
                       prevalence=~treatment + s(pid_rep),
                       data=meta, runs=20)

out <- multiSTM(mod.out, mass.threshold = .75,
                reg.formula = ~ treatment,
                metadata = gadarian)
plot(out)

# Same example as above, but loading from disk
mod.out <- selectModel(docs, vocab, K=3,
                       prevalence=~treatment + s(pid_rep),
                       data=meta, runs=20, to.disk=T)

out <- multiSTM(from.disk=T, mass.threshold = .75,
                reg.formula = ~ treatment,
                metadata = gadarian)

## End(Not run)
```

`optimizeDocument` *Optimize Document*

Description

A primarily internal use function for optimizing the document-level parameters of the variational distribution. Included here for advanced users who want to design new post-processing features. This help file assumes knowledge of our notation which follows the mathematical notation used in our vignette and other papers.

Usage

```
optimizeDocument(
  document,
  eta,
  mu,
  beta,
  sigma = NULL,
  sigmainv = NULL,
  sigmaentropy = NULL,
  method = "BFGS",
  control = list(maxit = 500),
  posterior = TRUE
)
```

Arguments

document	a single matrix containing the document in the stm format
eta	a vector of length K-1 containing the initial starting value for eta
mu	a vector of length K-1 containing the prevalence prior
beta	a matrix containing the complete topic-word distribution for the document. If using a content covariate model it is presumed that you have already passed the correct content covariate level's beta.
sigma	a K-1 by K-1 matrix containing the covariance matrix of the MVN prior. If you supply this you do not need to supply sigmainv or sigmaentropy. See below.
sigmainv	a K-1 by K-1 matrix containing the precision matrix of the MVN prior. If you supplied sigma you do not need to supply this. See below.
sigmaentropy	the entropy term calculated from sigma. If you supplied sigma you do not need to supply this. See below.
method	the method passed to optim. Uses "BFGS" by default.
control	the control argument passed to optim. Sets the maximum number of observations to 500 but can be used to set other aspects of the optimization per the instructions in optim
posterior	should the full posterior be returned? If TRUE (as it is by default) returns the full variational posterior. Otherwise just returns the point estimate.

Details

This function is a small wrapper around the internal function used to complete the E-step for each document.

Regarding the arguments sigma, sigmainv and sigmaentropy. In the internal version of the code we calculate sigmainv and sigmaentropy once each E-step because it is shared by all documents. If you supply the original value to sigma it will calculate these for you. If you are going to be using this to run a bunch of documents and speed is a concern, peek at the underlying code and do the calculation yourself once and then just pass the result to the function so it isn't repeated with every observation.

Value

a list

phis	A K by V* matrix containing the variational distribution for each token (where V* is the number of unique words in the given document. They are in the order of appearance in the document. For words repeated more than once the sum of the column is the number of times that token appeared.
lambda	A (K-1) by 1 matrix containing the mean of the variational distribution for eta. This is actually just called eta in the output of stm as it is also the point estimate.
nu	A (K-1) by (K-1) matrix containing the covariance matrix of the variational distribution for eta. This is also the inverse Hessian matrix.
bound	The value of the document-level contribution to the global approximate evidence lower bound.

See Also

thetaPosterior

Examples

```
# fitting to a nonsense word distribution
V <- length(poliblog5k.voc)
K <- 50
beta <- matrix(rgamma(V*K,shape = .1), nrow=K, ncol=V)
beta <- beta/rowSums(beta)
doc <- poliblog5k.docs[[1]]
mu <- rep(0, K-1)
sigma <- diag(1000, nrow=K-1)
optimizeDocument(doc, eta=rep(0, K-1), mu=mu, beta=beta, sigma=sigma)
```

---

permutationTest *Permutation test of a binary covariate.*

Description

Run a permutation test where a binary treatment variable is randomly permuted and topic model is reestimated.

Usage

```
permutationTest(
  formula,
  stmobj,
  treatment,
  nruns = 100,
  documents,
  vocab,
  data,
  seed = NULL,
  stmverbose = TRUE,
  uncertainty = "Global"
)
```

Arguments

formula	A formula for the prevalence component of the stm model and the estimateEffect call. This formula must contain at least one binary covariate (specified through the argument treatment) but it can contain other terms as well. If the binary covariate is interacted with additional variables the estimated quantity of interest is the effect when those additional variables are set to 0.
stmobj	Model output from a single run of stm which contains the reference effect.

treatment	A character string containing treatment id as used in the formula of the stmobj. This is the variable which is randomly permuted.
nruns	Number of total models to fit (including the original model).
documents	The documents used in the stmobj model.
vocab	The vocab used in the stmobj model.
data	The data used in the stmobj model.
seed	Optionally a seed with which to replicate the result. As in stm the seed is automatically saved and returned as part of the object. Passing the seed here will replicate the previous run.
stmverbose	Should the stm model be run with verbose=TRUE. Turning this to FALSE will suppress only the model specific printing. An update on which model is being run will still print to the screen.
uncertainty	Which procedure should be used to approximate the measurement uncertainty in the topic proportions. See details for more information. Defaults to the Global approximation.

Details

This function takes a single binary covariate and runs a permutation test where, rather than using the true assignment, the covariate is randomly drawn with probability equal to its empirical probability in the data. After each shuffle of the covariate the same STM model is estimated at different starting values using the same initialization procedure as the original model, and the effect of the covariate across topics is calculated.

Next the function records two quantities of interest across this set of "runs" of the model. The first records the absolute maximum effect of the permuted covariate across all topics.

The second records the effect of the (permuted) covariate on the topic in each additional stm run which is estimated to be the topic closest to the topic of interest (specified in plot.STMpermute) from the original stm model. Uncertainty can be calculated using the standard options in estimateEffect.

Value

ref	A list of K elements containing the quantiles of the estimated effect for the reference model.
permute	A list where each element is an aligned model parameter summary
variable	The variable id that was permuted.
seed	The seed for the stm model.

See Also

plot.STMpermute

Examples

```
## Not run:
temp<-textProcessor(documents=gadarian$open.ended.response,metadata=gadarian)
out <- prepDocuments(temp$documents, temp$vocab, temp$meta)
documents <- out$documents
vocab <- out$vocab
meta <- out$meta
set.seed(02138)
mod.out <- stm(documents, vocab, 3, prevalence=~treatment + s(pid_rep), data=meta)
summary(mod.out)
prep <- estimateEffect(1:3 ~ treatment + s(pid_rep), mod.out, meta)
plot(prep, "treatment", model=mod.out,
     method="difference",cov.value1=1,cov.value2=0)
test <- permutationTest(formula=~ treatment + s(pid_rep), stmobj=mod.out,
                        treatment="treatment", nruns=25, documents=documents,
                        vocab=vocab,data=meta, stmverbose=FALSE)
plot(test,2, xlab="Effect", ylab="Model Index", main="Topic 2 Placebo Test")

## End(Not run)
```

`plot.estimateEffect` *Plot effect of covariates on topics*

Description

Plots the effect of a covariate on a set of topics selected by the user. Different effect types available depending on type of covariate. Before running this, the user should run a function to simulate the necessary confidence intervals. See `estimateEffect`.

Usage

```
## S3 method for class 'estimateEffect'
plot(
  x,
  covariate,
  model = NULL,
  topics = x$topics,
  method = c("pointestimate", "difference", "continuous"),
  cov.value1 = NULL,
  cov.value2 = NULL,
  moderator = NULL,
  moderator.value = NULL,
  npoints = 100,
  nsims = 100,
  ci.level = 0.95,
  xlim = NULL,
  ylim = NULL,
```

```
  xlab = "",
  ylab = NULL,
  main = "",
  printlegend = T,
  labeltype = "numbers",
  n = 7,
  frexw = 0.5,
  add = F,
  linecol = NULL,
  width = 25,
  verbose.labels = T,
  family = NULL,
  custom.labels = NULL,
  omit.plot = FALSE,
  ...
)
```

Arguments

x	Output of estimateEffect, which calculates simulated betas for plotting.
covariate	String of the name of the main covariate of interest. Must be enclosed in quotes. All other covariates within the formula specified in estimateEffect will be kept at their median.
model	Model output, only necessary if labeltype is "prob", "frex", "score", or "lift". Models with more than one spline cannot be used for plot.estimateEffect.
topics	Topics to plot.
method	Method used for plotting. "pointestimate" estimates mean topic proportions for each value of the covariate. "difference" estimates the mean difference in topic proportions for two different values of the covariate (cov.value1 and cov.value2 must be specified). "continuous" estimates how topic proportions vary over the support of a continuous covariate.
cov.value1	For method "difference", the value or set of values of interest at which to set the covariate. In the case of calculating a treatment/control contrast, set the treatment to cov.value1.
cov.value2	For method "difference", the value or set of values which will be set as the comparison group. cov.value1 and cov.value2 must be vectors of the same length.
moderator	When two terms are interacted and one variable in the interaction is the covariate of interest, the user can specify the value of the interaction with moderator.value, and the name of the moderator with moderator.
moderator.value	When two terms are interacted and one variable in the interaction is the covariate of interest, the user can specify the value of the interaction term.
npoints	Number of unique points to use for simulation along the support of a continuous covariate. For method "continuous" only.
nsims	Number of simulations for estimation.
ci.level	Confidence level for confidence intervals.

xlim	Vector of x axis minimum and maximum values.
ylim	Vector of y axis minimum and maximum values.
xlab	Character string that is x axis title.
ylab	Character string that is y axis title.
main	Character string that is plot title.
printlegend	Whether to plot a topic legend in the case of a continuous covariate.
labeltype	Determines the labeltype for the topics. The default is "number" which prints the topic number. Other options are "prob", which prints the highest probability words, "score", "lift", and "frex", from labeltopics (see labeltopics() for more details). The user can also select "custom" for custom labels, which should be inputted under custom.labels. Labels appear in the legend for continuous covariates.
n	Number of words to print if "prob", "score", "lift", or "frex" is chosen.
frexw	If "frex" labeltype is used, this will be the frex weight.
add	Logical parameter for whether the line should be added to the plot, or a new plot should be drawn.
linecol	For continuous covariates only. A vector that specifies the colors of the lines within the plot. If NULL, then colors will be randomly generated.
width	Number that specifies width of the character string. Smaller numbers will have smaller-width labels. Default is 25.
verbose.labels	For method "difference" – verboselabels will specify the comparison covariate values of the covariate on the plot.
family	Font family.
custom.labels	A vector of custom labels if labeltype is equal to "custom".
omit.plot	Defaults to FALSE. When set to TRUE returns everything invisibly but doesn't do any plotting.
...	Other plotting parameters

Value

Values returned invisibly will depend on the method

For pointestimate:

uvals	Values of the covariate at which means and ci's were evaluated.
topics	Topics for which means and ci's were evaluated.
means	For each topic, means for each unique value.
cis	For each topic, confidence intervals for each unique value.
labels	Labels for each topic and unique value.

For difference:

topics	Topics for which difference in means and ci's were evaluated
means	For each topic, difference in means.

cis	For each topic, confidence intervals for difference in means.
labels	Labels for each topic.

For continuous:

x	Individual values of the covariate at which means and ci's were evaluated.
topics	Topics for which means and ci's were evaluated
means	For each topic and each x, means.
cis	For each topic and each x, confidence intervals for difference in means.
labels	Labels for each topic.

Examples

```
prep <- estimateEffect(1:3 ~ treatment, gadarianFit, gadarian)
plot(prep, "treatment", model=gadarianFit,
method="pointestimate")
plot(prep, "treatment", model=gadarianFit,
method="difference",cov.value1=1,cov.value2=0)

#If the covariate were a binary factor,
#the factor labels can be used to
#specify the values of cov.value1 (e.g., cov.value1="treat").

# String variables must be turned to factors prior to plotting.
#If you see this error, Error in rep.int(c(1, numeric(n)), n - 1L) :
# invalid 'times' value, then you likely have not done this.

#Example of binary times binary interaction
gadarian$binaryvar <- sample(c(0,1), nrow(gadarian), replace=TRUE)
temp <- textProcessor(gadarian$open.ended.response,metadata=gadarian)
out <- prepDocuments(temp$documents, temp$vocab, temp$meta)
stm1 <- stm(out$documents, out$vocab, 3, prevalence=~treatment*binaryvar,
 data=gadarian)
prep <- estimateEffect(c(2) ~ treatment*binaryvar, stmobj=stm1,
metadata=gadarian)

par(mfrow=c(1,2))
plot(prep, "treatment", method="pointestimate",
cov.value1=1, cov.value2=0, xlim=c(-1,1), moderator="binaryvar", moderator.value=1)
plot(prep, "treatment", method="pointestimate",
cov.value1=1, cov.value2=0, xlim=c(-1,1), moderator="binaryvar",
moderator.value=0)
```

plot.MultimodDiagnostic *Plotting Method for Multimodality Diagnostic Objects*

Description

The plotting method for objects of the S3 class 'MultimodDiagnostic', which are returned by the function `multiSTM()`, which performs a battery of tests aimed at assessing the stability of the local modes of an STM model.

Usage

```
## S3 method for class 'MultimodDiagnostic'
plot(x, ind = NULL, topics = NULL, ...)
```

Arguments

`x`	An object of S3 class 'MultimodDiagnostic'. See `multiSTM`.
`ind`	An integer of list of integers specifying which plots to generate (see details). If `NULL` (default), all plots are generated.
`topics`	An integer or vector of integers specifying the topics for which to plot the posterior distribution of covariate effect estimates. If `NULL` (default), plots are generated for every topic in the S3 object.
`...`	Other arguments to be passed to the plotting functions.

Details

This methods generates a series of plots, which are indexed as follows. If a subset of the plots is required, specify their indexes using the `ind` argument. Please note that not all plot types are available for every object of class 'MultimodDiagnostic':

1. Histogram of Expected Common Words: Generates a 10-bin histogram of the column means of `obj$wmat`, a K-by-N matrix reporting the number of "top words" shared by the reference model and the candidate model. The "top words" for a given topic are defined as the 10 highest-frequency words.
2. Histogram of Expected Common Documents: Generates a 10-bin histogram of the column means of `obj$tmat`, a K-by-N matrix reporting the number of "top documents" shared by the reference model and the candidate model. The "top documents" for a given topic are defined as the 10 documents in the reference corpus with highest topical frequency.
3. Distribution of .95 Confidence-Interval Coverage for Regression Estimates: Generates a histogram of `obj$confidence.ratings`, a vector whose entries specify the proportion of regression coefficient estimates in a candidate model that fall within the .95 confidence interval for the corresponding estimate in the reference model. This can only be generated if `obj$confidence.ratings` is non-`NULL`.

4. Posterior Distributions of Covariate Effect Estimates By Topic: Generates a square matrix of plots, each depicting the posterior distribution of the regression coefficients for the covariate specified in `obj$reg.parameter.index` for one topic. The topics for which the plots are to be generated are specified by the `topics` argument. If the length of `topics` is not a perfect square, the plots matrix will include white space. The plots have a dashed black vertical line at zero, and a continuous red vertical line indicating the coefficient estimate in the reference model. This can only be generated if `obj$cov.effects` is non-`NULL`.
5. Histogram of Expected L1-Distance From Reference Model: Generates a 10-bin histogram of the column means of `obj$lmat`, a K-by-N matrix reporting the L1-distance of each topic from the corresponding one in the reference model.
6. L1-distance vs. Top-10 Word Metric: Produces a smoothed color density representation of the scatterplot of `obj$lmat` and `obj$wmat`, the metrics for L1-distance and shared top-words, obtained through a kernel density estimate. This can be used to validate the metrics under consideration.
7. L1-distance vs. Top-10 Docs Metric: Produces a smoothed color density representation of the scatterplot of `obj$lmat` and `obj$tmat`, the metrics for L1-distance and shared top-documents, obtained through a kernel density estimate. This can be used to validate the metrics under consideration.
8. Top-10 Words vs. Top-10 Docs Metric: Produces a smoothed color density representation of the scatterplot of `obj$wmat` and `obj$tmat`, the metrics for shared top-words and shared top-documents, obtained through a kernel density estimate. This can be used to validate the metrics under consideration.
9. Maximized Bound vs. Aggregate Top-10 Words Metric: Generates a scatter plot with linear trendline for the maximized bound vector (`obj$lb`) and a linear transformation of the top-words metric aggregated by model (`obj$wmod/1000`).
10. Maximized Bound vs. Aggregate Top-10 Docs Metric: Generates a scatter plot with linear trendline for the maximized bound vector (`obj$lb`) and a linear transformation of the top-docs metric aggregated by model (`obj$tmod/1000`).
11. Maximized Bound vs. Aggregate L1-Distance Metric: Generates a scatter plot with linear trendline for the maximized bound vector (`obj$lb`) and a linear transformation of the L1-distance metric aggregated by model (`obj$tmod/1000`).
12. Top-10 Docs Metric vs. Semantic Coherence: Generates a scatter plot with linear trendline for the reference-model semantic coherence scores and the column means of `object$tmat`.
13. L1-Distance Metric vs. Semantic Coherence: Generates a scatter plot with linear trendline for the reference-model semantic coherence scores and the column means of `object$lmat`.
14. Top-10 Words Metric vs. Semantic Coherence: Generates a scatter plot with linear trendline for the reference-model semantic coherence scores and the column means of `object$wmat`.
15. Same as 5, but using the limited-mass L1-distance metric. Can only be generated if `obj$mass.threshold != 1`.
16. Same as 11, but using the limited-mass L1-distance metric. Can only be generated if `obj$mass.threshold != 1`.
17. Same as 7, but using the limited-mass L1-distance metric. Can only be generated if `obj$mass.threshold != 1`.
18. Same as 13, but using the limited-mass L1-distance metric. Can only be generated if `obj$mass.threshold != 1`.

Author(s)

Brandon M. Stewart (Princeton University) and Antonio Coppola (Harvard University)

References

Roberts, M., Stewart, B., & Tingley, D. (Forthcoming). "Navigating the Local Modes of Big Data: The Case of Topic Models. In Data Analytics in Social Science, Government, and Industry." New York: Cambridge University Press.

See Also

multiSTM

Examples

```
## Not run:

# Example using Gadarian data

temp<-textProcessor(documents=gadarian$open.ended.response,
                    metadata=gadarian)
meta<-temp$meta
vocab<-temp$vocab
docs<-temp$documents
out <- prepDocuments(docs, vocab, meta)
docs<-out$documents
vocab<-out$vocab
meta <-out$meta
set.seed(02138)
mod.out <- selectModel(docs, vocab, K=3,
                       prevalence=~treatment + s(pid_rep),
                       data=meta, runs=20)

out <- multiSTM(mod.out, mass.threshold = .75,
                reg.formula = ~ treatment,
                metadata = gadarian)

plot(out)
plot(out, 1)

## End(Not run)
```

---

plot.searchK *Plots diagnostic values resulting from searchK*

Description

Takes the result of searchK and produces a set of plots for evaluating optimal topic numbers via visual representation of diagnostic functions.

Usage

```
## S3 method for class 'searchK'
plot(x, ...)
```

Arguments

x	A searchK object, containing the diagnostic information of an stm with a variety of topics.
...	additional arguments for S3 compatibility.

Examples

```
K<-c(5,10,15)
temp<-textProcessor(documents=gadarian$open.ended.response,metadata=gadarian)
out <- prepDocuments(temp$documents, temp$vocab, temp$meta)
documents <- out$documents
vocab <- out$vocab
meta <- out$meta
set.seed(02138)
K<-c(5,10,15)
kresult <- searchK(documents, vocab, K, prevalence=~treatment + s(pid_rep), data=meta)

plot(kresult)
```

`plot.STM` *Functions for plotting STM objects*

Description

Produces one of four types of plots for an STM object. The default option `"summary"` prints topic words with their corpus frequency. `"labels"` is for easy printing of tables of indicative words for each topic. `"perspectives"` depicts differences between two topics, content covariates or combinations. `"hist"` creates a histogram of the expected distribution of topic proportions across the documents.

Usage

```
## S3 method for class 'STM'
plot(
  x,
  type = c("summary", "labels", "perspectives", "hist"),
  n = NULL,
  topics = NULL,
  labeltype = c("prob", "frex", "lift", "score"),
  frexw = 0.5,
  main = NULL,
  xlim = NULL,
  ylim = NULL,
  xlab = NULL,
  family = "",
  width = 80,
  covarlevels = NULL,
  plabels = NULL,
  text.cex = 1,
  custom.labels = NULL,
  topic.names = NULL,
  ...
)
```

Arguments

x	Model output from stm.
type	Sets the desired type of plot. See details for more information.
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. The defaults are 3, 20 and 25 for `summary`, `labels` and `perspectives` respectively. n must be greater than or equal to 2
topics	Vector of topics to display. For plot perspectives this must be a vector of length one or two. For the other two types it defaults to all topics.
labeltype	Determines which option of `"prob"`, `"frex"`, `"lift"`,`"score"` is used for choosing the most important words. See `labelTopics` for more detail. Passing an argument to `custom.labels` will override this. Note that this does not apply to `perspectives` type which always uses highest probability words.
frexw	If "frex" labeltype is used, this will be the frex weight.
main	Title to the plot
xlim	Range of the X-axis.
ylim	Range of the Y-axis.
xlab	Labels for the X-axis. For perspective plots, use `plabels` instead.
family	The Font family. Most of the time the user will not need to specify this but if using other character sets can be useful see par.
width	Sets the width in number of characters used for string wrapping in type `"labels"`

covarlevels	A vector of length one or length two which contains the levels of the content covariate to be used in perspective plots.
plabels	This option can be used to override the default labels in the perspective plot that appear along the x-axis. It should be a character vector of length two which has the left hand side label first.
text.cex	Controls the scaling constant on text size.
custom.labels	A vector of custom labels if labeltype is equal to "custom".
topic.names	A vector of custom topic names. Defaults to "Topic #: ".
...	Additional parameters passed to plotting functions.

Details

The function can produce three types of plots which summarize an STM object which is chosen by the argument `type`. `summary` produces a plot which displays the topics ordered by their expected frequency across the corpus. `labels` plots the top words selected according to the chosen criteria for each selected topics. `perspectives` plots two topic or topic-covariate combinations. Words are sized proportional to their use within the plotted topic-covariate combinations and oriented along the X-axis based on how much they favor one of the two configurations. If the words cluster on top of each other the user can either set the plot size to be larger or shrink the total number of words on the plot. The vertical configuration of the words is random and thus can be rerun to produce different results each time. Note that `perspectives` plots do not use any of the labeling options directly. `hist` plots a histogram of the MAP estimates of the document-topic loadings across all documents. The median is also denoted by a dashed red line.

References

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.

See Also

plotQuote, plot.topicCorr

Examples

```
#Examples with the Gadarian Data
plot(gadarianFit)
plot(gadarianFit,type="labels")
plot(gadarianFit, type="perspectives", topics=c(1,2))
plot(gadarianFit,type="hist")
```

`plot.STMpermute` *Plot an STM permutation test.*

Description

Plots the results of a permutation test run using `permutationTest`.

Usage

```
## S3 method for class 'STMpermute'
plot(x, topic, type = c("match", "largest"), xlim = NULL, ylim = NULL, ...)
```

Arguments

`x`	Object from the output of `permutationTest`.
`topic`	Integer indicating which topic to plot.
`type`	Character string indicating what topic comparison to use. "match" uses the Hungarian aligned method and "largest" uses the largest mean in direction of reference topic.
`xlim`	Range of the X-axis.
`ylim`	Range of the Y-axis.
`...`	Other parameters which may be passed to plot.

Details

This function plots the output of `permutationTest` by stacking horizontal confidence intervals for the effects of the permuted variable. In choosing the topic in the permuted runs of stm to plot the effect for, two methods are available, "match" and "largest". The former uses Kuhn's (1955) Hungarian method to align the topics, and then uses the model's best match of the reference topic. The latter uses the topic which has the expected effect size in the direction of the reference model effect; thus, we would expect this method to be quite conservative.

See Also

`permutationTest`

Examples

```
## Not run:

temp<-textProcessor(documents=gadarian$open.ended.response,metadata=gadarian)
out <- prepDocuments(temp$documents, temp$vocab, temp$meta)
documents <- out$documents
vocab <- out$vocab
meta <- out$meta
set.seed(02138)
```

```
mod.out <- stm(documents, vocab, 3, prevalence=~treatment + s(pid_rep), data=meta)
summary(mod.out)
prep <- estimateEffect(1:3 ~ treatment + s(pid_rep), mod.out, meta)
plot(prep, "treatment", model=mod.out,
     method="difference",cov.value1=1,cov.value2=0)
test <- permutationTest(formula=~ treatment + s(pid_rep), stmobj=mod.out,
                        treatment="treatment", nruns=25, documents=documents,
                        vocab=vocab,data=meta, stmverbose=FALSE)
plot(test,2, xlab="Effect", ylab="Model Index", main="Topic 2 Placebo Test")

## End(Not run)
```

`plot.topicCorr` *Plot a topic correlation graph*

Description

Uses a topic correlation graph estimated by `topicCorr` and the `igraph` package to plot a network where nodes are topics and edges indicate a positive correlation.

Usage

```
## S3 method for class 'topicCorr'
plot(
  x,
  topics = NULL,
  vlabels = NULL,
  layout = NULL,
  vertex.color = "green",
  vertex.label.cex = 0.75,
  vertex.label.color = "black",
  vertex.size = NULL,
  ...
)
```

Arguments

`x`	A topicCorr model object.
`topics`	A vector of topics to include in the plot, defaults to all.
`vlabels`	A character vector of labels for the vertices. Defaults to "Topic #"
`layout`	The layout algorithm passed to the `igraph` package. It will choose `layout.fruchterman.reingold` by default. Note that to pass an alternate algorithm you should load the `igraph` package first.
`vertex.color`	Color of the vertices.
`vertex.label.cex`	Controls the size of the labels.

vertex.label.color
: Controls the color of the labels.

vertex.size
: Controls the sizes of the vertices, either NULL, a scalar or a vector of the same length as number of topics.

...
: Additional parameters passed to `plot.graph.adjacency`

Details

Essentially a thin wrapper around the plotting functionality in the `igraph` package. See package vignette for more details.

References

Csardi G, Nepusz T: The igraph software package for complex network research, InterJournal, Complex Systems 1695. 2006. http://igraph.sf.net

See Also

topicCorr

Examples

```
#This function becomes more useful with larger numbers of topics.
#it is demonstrated here with a small model simply to show how the syntax works.
cormat <- topicCorr(gadarianFit)
plot(cormat)
```

---

plotModels *Plots semantic coherence and exclusivity for high likelihood models outputted from selectModel.*

Description

Plots semantic coherence and exclusivity for high likelihood models. In the case of models that include content covariates, prints semantic coherence and sparsity.

Usage

```
plotModels(
  models,
  xlab = "Semantic Coherence",
  ylab = "Exclusivity",
  labels = 1:length(models$runout),
  pch = NULL,
```

```
  legend.position = "topleft",
  ...
)
```

Arguments

models	Output from selectModel.
xlab	Character string that is x axis title. This will be semantic coherence.
ylab	Character string that is y axis title. This will be exclusivity.
labels	Labels for each model.
pch	A vector of integers specifying symbol for plotting.
legend.position	The location of the legend. Can be "bottomright","bottom", "bottomleft", "left", "topleft", "top", "topright", "right" and "center".
...	Other plotting parameters.

Details

Each model has semantic coherence and exclusivity values associated with each topic. In the default plot function, the small colored dots are associated with a topic's semantic coherence and exclusivity. Dots with the same color as topics associated with the same model. The average semantic coherence and exclusivity is also plotted in the same color, but printed as the model number associated with the output from selectModels().

With content covariates, the model does not output exclusivity because exclusivity has been built in with the content covariates. Instead, the user should check to make sure that sparsity is high enough (typically greater than .5), and then should select a model based on semantic coherence.

---

plotQuote *Plots strings*

Description

Plots strings to a blank canvas. Used primarily for plotting quotes generated by `findThoughts`.

Usage

```
plotQuote(
  sentences,
  width = 30,
  text.cex = 1,
  maxwidth = NULL,
  main = NULL,
  xlab = "",
  ylab = "",
  xlim = NULL,
  ylim = NULL,
  ...
)
```

Arguments

sentences	Vector of sentence to plot.
width	Number of characters in each line.
text.cex	Sets the size of the text
maxwidth	Sets the maximum character width of the plotted responses rounding to the nearest word. Note that this may perform somewhat unexpectedly for very small numbers.
main	Title of plot.
xlab	Sets an x-axis label
ylab	Set a y-axis label
xlim	Sets the x-range of the plot.
ylim	Sets the y-range of the plot
...	Other parameters passed to the plot function

Details

A simple function which wraps sentences at width characters per line and plots the results.

See Also

findThoughts

Examples

```
thoughts <- findThoughts(gadarianFit,texts=gadarian$open.ended.response,
topics=c(1), n=3)$docs[[1]]
plotQuote(thoughts)
```

---

plotRemoved *Plot documents, words and tokens removed at various word thresholds*

Description

A plot function which shows the results of using different thresholds in prepDocuments on the size of the corpus.

Usage

```
plotRemoved(documents, lower.thresh)
```

Arguments

documents	The documents to be used for the stm model
lower.thresh	A vector of integers, each of which will be tested as a lower threshold for the prepDocuments function.

Details

For a lower threshold, prepDocuments will drop words which appear in fewer than that number of documents, and remove documents which contain no more words. This function allows the user to pass a vector of lower thresholds and observe how prepDocuments will handle each threshold. This function produces three plots, showing the number of words, the number of documents, and the total number of tokens removed as a function of threshold values. A dashed red line is plotted at the total number of documents, words and tokens respectively.

Value

Invisibly returns a list of

lower.thresh	The sorted threshold values
ndocs	The number of documents dropped for each value of the lower threshold
nwords	The number of entries of the vocab dropped for each value of the lower threshold.
ntokens	The number of tokens dropped for each value of the lower threshold.

See Also

prepDocuments

Examples

```
plotRemoved(poliblog5k.docs, lower.thresh=seq(from = 10, to = 1000, by = 10))
```

---

plotTopicLoess *Plot some effects with loess*

Description

Plots a loess line of the topic proportions on a covariate inputted by the user. This allows for a more flexible functional form for the relationship.

Usage

```
plotTopicLoess(
  model,
  topics,
  covariate,
  span = 1.5,
  level = 0.95,
  main = "",
  xlab = "Covariate",
  ylab = "Topic Proportions"
)
```

Arguments

model	An STM model object
topics	Vector of topic numbers to plot by the covariate. E.g., c(1,2,3) would plot lines for topics 1,2,3.
covariate	Covariate vector by which to plot topic proportions.
span	loess span parameter. See loess
level	Desired coverage for confidence intervals
main	Title of the plot, default is ""
xlab	X-label, default is "Covariate"
ylab	Y-label, default is "Topic Proportions"

Details

This function is considerably less developed than plot.estimateEffect and we recommend using that function with splines and high degrees of freedom where possible. Computes standard errors through the method of composition as in estimateEffect.

See Also

plot.estimateEffect

Examples

```
plotTopicLoess(gadarianFit, topics=1, covariate=gadarian$pid_rep)
```

poliblog5k *CMU 2008 Political Blog Corpus*

Description

A 5000 document sample from CMU 2008 Political Blog Corpus (Eisenstein and Xing 2010). Blog posts from 6 blogs during the U.S. 2008 Presidential Election.

Format

A data frame with 5000 observations on the following 4 variables.

`rating` a factor variable giving the partisan affiliation of the blog (based on who they supported for president)

`day` the day of the year (1 to 365). All entries are from 2008.

`blog` a two digit character code corresponding to the name of the blog. They are: American Thinker (at), Digby (db), Hot Air (ha), Michelle Malkin (mm), Think Progress (tp), Talking Points Memo (tpm)

`text` the first 50 characters (rounded to the nearest full word).

Details

This is a random sample of the larger CMU 2008 Political Blog Corpus collected by Jacob Eisenstein and Eric Xing. Quoting from their documentation: "[The blogs] were selected by the following criteria: the Technorati rankings of blog authority, ideological balance, coverage for the full year 2008, and ease of access to blog archives. In the general election for U.S. President in 2008, the following blogs supported Barack Obama: Digby, ThinkProgress, and Talking Points Memo. John McCain was supported by American Thinker, Hot Air, and Michelle Malkin. In general, the blogs that supported Obama in the election tend to advocate for similar policies and candidates as the Democratic party; and the blogs that supported McCain tend to advocate Republican policies and candidates. Digby, Hot Air and Michelle Malkin are single-author blogs; the others have multiple authors."

Source

Jacob Eisenstein and Eric Xing (2010) "The CMU 2008 Political Blog Corpus." Technical Report Carnegie Mellon University. http://sailing.cs.cmu.edu/socialmedia/blog2008.html

Examples

```
data(poliblog5k)
head(poliblog5k.meta)
head(poliblog5k.voc)
```

```
stm1 <- stm(poliblog5k.docs, poliblog5k.voc, 3,
prevalence=~rating, data=poliblog5k.meta)
```

`prepDocuments` *Prepare documents for analysis with* `stm`

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).

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 `stm`.
`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 at 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.

The behavior is such that when `prepDocuments` drops documents their corresponding rows are deleted and the row names are not renumbered. We however do not recommend using rownames for joins- instead the best practice is to either keep a unique identifier in the `meta` object for doing joins or use something like **quanteda** which has a more robust interface for manipulating the corpus itself.

If you have any documents which are of length 0 in your original object the function will throw an error. These should be removed before running the function although please be sure to remove the corresponding rows in the meta data file if you have one. You can quickly identify the documents using the code: `which(unlist(lapply(documents, length))==0)`.

Value

A list containing a new documents and vocab object.

`documents`	The new documents object for use with `stm`
`vocab`	The new vocab object for use with `stm`
`meta`	The new meta data object for use with `stm`. Will be the same if no documents are removed.
`words.removed`	A set of indices corresponding to the positions in the original vocab object of words which have been removed.
`docs.removed`	A set of indices corresponding to the positions in the original documents object of documents which no longer contained any words after dropping terms from the vocab.
`tokens.removed`	An integer corresponding to the number of unique tokens removed from the corpus.
`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

`plotRemoved`

Examples

```
temp<-textProcessor(documents=gadarian$open.ended.response,metadata=gadarian)
meta<-temp$meta
```

```
vocab<-temp$vocab
docs<-temp$documents
out <- prepDocuments(docs, vocab, meta)
```

`readCorpus` *Read in a corpus file.*

Description

Converts pre-processed document matrices stored in popular formats to stm format.

Usage

```
readCorpus(corpus, type = c("dtm", "slam", "Matrix"))
```

Arguments

`corpus`	An input file or filepath to be processed
`type`	The type of input file. We offer several sources, see details.

Details

This function provides a simple utility for converting other document formats to our own. Briefly- `dtm` takes as input a standard matrix and converts to our format. `slam` converts from the `simple_triplet_matrix` representation used by the `slam` package. This is also the representation of corpora in the popular `tm` package and should work in those cases.

`dtm` expects a matrix object where each row represents a document and each column represents a word in the dictionary.

`slam` expects a `simple_triplet_matrix` from that package.

`Matrix` attempts to coerce the matrix to a `simple_triplet_matrix` and convert using the functionality built for the `slam` package. This will work for most applicable classes in the `Matrix` package such as `dgCMatrix`.

If you are trying to read a `.ldac` file see `readLdac`.

Value

`documents`	A documents object in our format
`vocab`	A vocab object if information is available to construct one

See Also

`textProcessor`, `prepDocuments` `readLdac`

Examples

```
## Not run:

library(textir)
data(congress109)
out <- readCorpus(congress109Counts, type="Matrix")
documents <- out$documents
vocab <- out$vocab

## End(Not run)
```

`readLdac` *Read in a .ldac Formatted File*

Description

Read in a term document matrix in the .ldac sparse matrix format popularized by David Blei's C code implementation of lda.

Usage

```
readLdac(filename)
```

Arguments

`filename`	An input file or filepath to be processed

Details

`ldac` expects a file name or path that contains a file in Blei's LDA-C format. From his ReadMe: "The data is a file where each line is of the form:

[M] [term_1]:[count] [term_2]:[count] ... [term_N]:[count]

where [M] is the number of unique terms in the document, and the [count] associated with each term is how many times that term appeared in the document. Note that [term_1] is an integer which indexes the term; it is not a string."

Because R indexes from one, the values of the term indices are incremented by one on import.

Value

`documents`	A documents object in our format

See Also

`textProcessor`, `prepDocuments` `readCorpus`

s *Make a B-spline Basis Function*

Description

This is a simple wrapper around the bs function in the splines package. It will default to a spline with 10 degrees of freedom.

Usage

```
s(x, df, ...)
```

Arguments

x	The predictor value.
df	Degrees of freedom. Defaults to the minimum of 10 or one minus the number of unique values in x.
...	Arguments passed to the bs function.

Details

This is a simple wrapper written as users may find it easier to simply type s rather than selecting parameters for a spline. We also include predict and makepredictcall generic functions for the class so it will work in settings where predict is called.

Value

A predictor matrix of the basis functions.

See Also

bs ns

sageLabels *Displays verbose labels that describe topics and topic-covariate groups in depth.*

Description

For each topic or, when there is a covariate at the bottom of the model, for each topic-covariate group, sageLabels provides a list of the highest marginal probability words, the highest marginal FREX words, the highest marginal lift words, and the highest marginal score words, where marginal means it is summing over all potential covariates. It also provides each topic's Kappa (words associated with each topic) and baselined Kappa (baseline word distribution).

Usage

```
sageLabels(model, n = 7)
```

Arguments

model	A fitted STM model object.
n	The number of words to print per topic/topic-covariate set. Default is 7.

Details

This can be used as an more detailed alternative to labelTopics.

Value

marginal	A list of matrices, containing the high-probability labels, FREX labels, lift labels, and high scoring words.
K	The number of topics in the STM.
covnames	Names of the covariate values used in the STM.
kappa	Words associated with topics, covariates, and topic/covariate interactions.
kappa.m	Baseline word distribution.
n	The n parameter passed by the user to this function; number of words per topic or topic-covariate pair (when covariates are used on the bottom of the model)
cov.betas	Covariate-specific beta matrices, listing for each covariate a matrix of highest-probability, FREX, lift, and high scoring words. Note that the actual vocabulary has been substituted for word indices.

---

searchK *Computes diagnostic values for models with different values of K (number of topics).*

Description

With user-specified initialization, this function runs selectModel for different user-specified topic numbers and computes diagnostic properties for the returned model. These include exclusivity, semantic coherence, heldout likelihood, bound, lbound, and residual dispersion.

Usage

```
searchK(
  documents,
  vocab,
  K,
  init.type = "Spectral",
  N = floor(0.1 * length(documents)),
  proportion = 0.5,
```

```
  heldout.seed = NULL,
  M = 10,
  cores = 1,
  ...
)
```

Arguments

documents	The documents to be used for the stm model
vocab	The vocabulary to be used for the stmmodel
K	A vector of different topic numbers
init.type	The method of initialization. See stm for options. Note that the default option here is different from the main function.
N	Number of docs to be partially held out
proportion	Proportion of docs to be held out.
heldout.seed	If desired, a seed to use when holding out documents for later heldout likelihood computation
M	M value for exclusivity computation
cores	Number of CPUs to use for parallel computation
...	Other diagnostics parameters.

Details

See the vignette for interpretation of each of these measures. Each of these measures is also available in exported functions:

exclusivity exclusivity

semantic coherence semanticCoherence

heldout likelihood make.heldout and eval.heldout

bound calculated by stm accessible by max(model$convergence$bound)

lbound a correction to the bound that makes the bounds directly comparable max(model$convergence$bound) + lfactorial(model$settings$dim$K)

residual dispersion checkResiduals

Due to the need to calculate the heldout-likelihood N documents have proportion of the documents heldout at random. This means that even with the default spectral initialization the results can change from run to run. When the number of heldout documents is low or documents are very short, this also means that the results can be quite unstable. For example: the gadarian code demonstration below has heldout results based on only 34 documents and approximately 150 tokens total. Clearly this can lead to quite disparate results across runs. By contrast default settings for the poliblog5k dataset would yield a heldout sample of 500 documents with approximately 50000 tokens for the heldout sample. We should expect this to be substantially more stable.

Value

exclus	Exclusivity of each model.
semcoh	Semantic coherence of each model.
heldout	Heldout likelihood for each model.
residual	Residual for each model.
bound	Bound for each model.
lbound	lbound for each model.
em.its	Total number of EM iterations used in fiting the model.

See Also

plot.searchK make.heldout

Examples

```
K<-c(5,10,15)
temp<-textProcessor(documents=gadarian$open.ended.response,metadata=gadarian)
out <- prepDocuments(temp$documents, temp$vocab, temp$meta)
documents <- out$documents
vocab <- out$vocab
meta <- out$meta
set.seed(02138)
K<-c(5,10,15)
kresult <- searchK(documents, vocab, K, prevalence=~treatment + s(pid_rep), data=meta)
plot(kresult)
```

---

selectModel *Assists the user in selecting the best STM model.*

Description

Discards models with the low likelihood values based on a small number of EM iterations (cast net stage), then calculates semantic coherence, exclusivity, and sparsity (based on default STM run using selected convergence criteria) to allow the user to choose between models with high likelihood values.

Usage

```
selectModel(
  documents,
  vocab,
  K,
  prevalence = NULL,
  content = NULL,
  data = NULL,
  max.em.its = 100,
  verbose = TRUE,
  init.type = "LDA",
  emtol = 1e-05,
  seed = NULL,
  runs = 50,
  frexw = 0.7,
  net.max.em.its = 2,
  netverbose = FALSE,
  M = 10,
  N = NULL,
  to.disk = F,
  ...
)
```

Arguments

documents	The documents to be modeled. 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 similar to the format in the **lda** package except that (following R convention) the vocabulary is indexed from one. Corpora can be imported using the reader function and manipulated using the prepDocuments.
vocab	Character vector specifying the words in the corpus in the order of the vocab indices in documents. Each term in the vocabulary index must appear at least once in the documents. See prepDocuments for dropping unused items in the vocabulary.
K	A positive integer (of size 2 or greater) representing the desired number of topics. Additional detail on choosing the number of topics in details.
prevalence	A formula object with no response variable or a matrix containing topic prevalence covariates. Use s(), ns() or bs() to specify smooth terms. See details for more information.
content	A formula containing a single variable, a factor variable or something which can be coerced to a factor indicating the category of the content variable for each document.
data	Dataset which contains prevalence and content covariates.

max.em.its	The maximum number of EM iterations. If convergence has not been met at this point, a message will be printed.
verbose	A logical flag indicating whether information should be printed to the screen.
init.type	The method of initialization. Must be either Latent Dirichlet Allocation (LDA), Dirichlet Multinomial Regression Topic Model (DMR), a random initialization or a previous STM object.
emtol	Convergence tolerance. EM stops when the relative change in the approximate bound drops below this level. Defaults to .001%.
seed	Seed for the random number generator. stm saves the seed it uses on every run so that any result can be exactly reproduced. Setting the seed here simply ensures that the sequence of models will be exactly the same when respecified. Individual seeds can be retrieved from the component model objects.
runs	Total number of STM runs used in the cast net stage. Approximately 15 percent of these runs will be used for running a STM until convergence.
frexw	Weight used to calculate exclusivity
net.max.em.its	Maximum EM iterations used when casting the net
netverbose	Whether verbose should be used when calculating net models.
M	Number of words used to calculate semantic coherence and exclusivity. Defaults to 10.
N	Total number of models to retain in the end. Defaults to .2 of runs.
to.disk	Boolean. If TRUE, each model is saved to disk at the current directory in a separate RData file. This is most useful if one needs to run multiSTM() on a large number of output models.
...	Additional options described in details of stm.

Value

runout	List of model outputs the user has to choose from. Take the same form as the output from a stm model.
semcoh	Semantic coherence values for each topic within each model in runout
exclusivity	Exclusivity values for each topic within each model in runout. Only calculated for models without a content covariate
sparsity	Percent sparsity for the covariate and interaction kappas for models with a content covariate.

Examples

```
## Not run:

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

```
docs<-out$documents
vocab<-out$vocab
meta <-out$meta
set.seed(02138)
mod.out <- selectModel(docs, vocab, K=3, prevalence=~treatment + s(pid_rep),
                       data=meta, runs=5)
plotModels(mod.out)
selected<-mod.out$runout[[1]]

## End(Not run)
```

---

stm *Variational EM for the Structural Topic Model*

Description

Estimation of the Structural Topic Model using semi-collapsed variational EM. The function takes sparse representation of a document-term matrix, an integer number of topics, and covariates and returns fitted model parameters. Covariates can be used in the prior for topic `prevalence`, in the prior for topical `content` or both. See an overview of functions in the package here: `stm-package`

Usage

```
stm(
  documents,
  vocab,
  K,
  prevalence = NULL,
  content = NULL,
  data = NULL,
  init.type = c("Spectral", "LDA", "Random", "Custom"),
  seed = NULL,
  max.em.its = 500,
  emtol = 1e-05,
  verbose = TRUE,
  reportevery = 5,
  LDAbeta = TRUE,
  interactions = TRUE,
  ngroups = 1,
  model = NULL,
  gamma.prior = c("Pooled", "L1"),
  sigma.prior = 0,
  kappa.prior = c("L1", "Jeffreys"),
  control = list()
)
```

Arguments

documents	The document term matrix to be modeled. These can be supplied in the native **stm** format, a sparse term count matrix with one row per document and one column per term, or a **quanteda** dfm (document-feature matrix) object. When using the sparse matrix or quanteda format this will include the vocabulary and, for quanteda, optionally the metadata. If using the native list format, 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 similar to the format in the lda package except that (following R convention) the vocabulary is indexed from one. Corpora can be imported using the reader function and manipulated using the prepDocuments. Raw texts can be ingested using textProcessor. Note that when using **quanteda** dfm directly there may be higher memory use (because the texts and metadata are stored twice). You can convert from **quanteda**'s format directly to our native format using the **quanteda** function convert.
vocab	Character vector specifying the words in the corpus in the order of the vocab indices in documents. Each term in the vocabulary index must appear at least once in the documents. See prepDocuments for dropping unused items in the vocabulary. If documents is a sparse matrix or **quanteda** dfm object, then vocab should not (and must not) be supplied. It is contained already inside the column names of the matrix.
K	Typically a positive integer (of size 2 or greater) representing the desired number of topics. If init.type="Spectral" you can also set K=0 to use the algorithm of Lee and Mimno (2014) to set the number of topics (although unlike the standard spectral initialization this is not deterministic). Additional detail on choosing the number of topics below.
prevalence	A formula object with no response variable or a matrix containing topic prevalence covariates. Use s, ns or bs to specify smooth terms. See details for more information.
content	A formula containing a single variable, a factor variable or something which can be coerced to a factor indicating the category of the content variable for each document.
data	an optional data frame containing the prevalence and/or content covariates. If unspecified the variables are taken from the active environment.
init.type	The method of initialization, by default the spectral initialization. Must be either Latent Dirichlet Allocation ("LDA"), "Random", "Spectral" or "Custom". See details for more info. If you want to replicate a previous result, see the argument seed. For "Custom" see the format described below under the custom.beta option of the control parameters.
seed	Seed for the random number generator. stm saves the seed it uses on every run so that any result can be exactly reproduced. When attempting to reproduce a result with that seed, it should be specified here.
max.em.its	The maximum number of EM iterations. If convergence has not been met at this

point, a message will be printed. If you set this to 0 it will return the initialization.

`emtol` Convergence tolerance. EM stops when the relative change in the approximate bound drops below this level. Defaults to .00001. You can set it to 0 to have the algorithm run `max.em.its` number of steps. See advanced options under `control` for more options.

`verbose` A logical flag indicating whether information should be printed to the screen. During the E-step (iteration over documents) a dot will print each time 1% of the documents are completed. At the end of each iteration the approximate bound will also be printed.

`reportevery` An integer determining the intervals at which labels are printed to the screen during fitting. Defaults to every 5 iterations.

`LDAbeta` a logical that defaults to `TRUE` when there are no content covariates. When set to `FALSE` the model performs SAGE style topic updates (sparse deviations from a baseline).

`interactions` a logical that defaults to `TRUE`. This automatically includes interactions between content covariates and the latent topics. Setting it to `FALSE` reduces to a model with no interactive effects.

`ngroups` Number of groups for memoized inference. See details below.

`model` A prefit model object. By passing an `stm` object to this argument you can restart an existing model. See details for more info.

`gamma.prior` sets the prior estimation method for the prevalence covariate model. The default `Pooled` options uses Normal prior distributions with a topic-level pooled variance which is given a moderately regularizing half-cauchy(1,1) prior. The alternative `L1` uses `glmnet` to estimate a grouped penalty between L1-L2. If your code is running slowly immediately after "Completed E-Step" appears, you may want to switch to the `L1` option. See details below.

`sigma.prior` a scalar between 0 and 1 which defaults to 0. This sets the strength of regularization towards a diagonalized covariance matrix. Setting the value above 0 can be useful if topics are becoming too highly correlated.

`kappa.prior` sets the prior estimation for the content covariate coefficients. The default option is the `L1` prior. The second option is `Jeffreys` which is markedly less computationally efficient but is included for backwards compatibility. See details for more information on computation.

`control` a list of additional advanced parameters. See details.

Details

This is the main function for estimating a Structural Topic Model (STM). STM is an admixture with covariates in both mixture components. 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 K topics. These proportions are found in the N (number of documents) by K (user specified number of topics) theta matrix. Each of the K topics are represented as distributions over words. The K-by-V (number of words in the vocabulary) matrix logbeta contains the natural log of the probability of seeing each word conditional on the topic.

The most important user input in parametric topic models is the number of topics. There is no right answer to the appropriate number of topics. More topics will give more fine-grained representations of the data at the potential cost of being less precisely estimated. The number must be at least 2 which is equivalent to a unidimensional scaling model. For short corpora focused on very specific subject matter (such as survey experiments) 3-10 topics is a useful starting range. For small corpora (a few hundred to a few thousand) 5-50 topics is a good place to start. Beyond these rough guidelines it is application specific. Previous applications in political science with medium sized corpora (10k to 100k documents) have found 60-100 topics to work well. For larger corpora 100 topics is a useful default size. Of course, your mileage may vary.

When `init.type="Spectral"` and `K=0` the number of topics is set using the algorithm in Lee and Mimno (2014). See vignette for details. We emphasize here as we do there that this does not estimate the "true" number of topics and does not necessarily have any particular statistical properties for consistently estimating the number of topics. It can however provide a useful starting point.

The model for topical prevalence includes covariates which the analyst believes may influence the frequency with which a topic is discussed. This is specified as a formula which can contain smooth terms using splines or by using the function `s`. The response portion of the formula should be left blank. See the examples. These variables can include numeric and factor variables. While including variables of class `Dates` or other non-numeric, non-factor types will work in `stm` it may not always work for downstream functions such as `estimateEffect`.

The topical convent covariates are those which affect the way in which a topic is discussed. As currently implemented this must be a single variable which defines a discrete partition of the dataset (each document is in one and only one group). We may relax this in the future. While including more covariates in topical prevalence will rarely affect the speed of the model, including additional levels of the content covariates can make the model much slower to converge. This is due to the model operating in the much higher dimensional space of words in dictionary (which tend to be in the thousands) as opposed to topics.

In addition to the default priors for prevalence, we also make use of the `glmnet` package to allow for penalties between the L1 and L2 norm. In these settings we estimate a regularization path and then select the optimal shrinkage parameter using a user-tuneable information criterion. By default selecting the `L1` option will apply the L1 penalty selecting the optimal shrinkage parameter using AIC. The defaults have been specifically tuned for the STM but almost all the relevant arguments can be changed through the control structure below. Changing the `gamma.enet` parameters allow the user to choose a mix between the L1 and L2 norms. When set to 1 (as by default) this is the lasso penalty, when set to 0 its the ridge penalty. Any value in between is a mixture called the elastic net.

The default prior choice for content covariates is now the `L1` option. This uses an approximation framework developed in Taddy (2013) called Distributed Multinomial Regression which utilizes a factorized poisson approximation to the multinomial. See Roberts, Stewart and Airoldi (2014) for details on the implementation here. This is dramatically faster than previous versions. The old default setting which uses a Jeffreys prior is also available.

The argument `init.type` allows the user to specify an initialization method. The default choice, `"Spectral"`, provides a deterministic initialization using the spectral algorithm given in Arora et al 2014. See Roberts, Stewart and Tingley (2016) for details and a comparison of different approaches. Particularly when the number of documents is relatively large we highly recommend the Spectral algorithm which often performs extremely well. Note that the random seed plays no role in the spectral initialization as it is completely deterministic (unless using the `K=0` or random projection settings). When the vocab is larger than 10000 terms we use only the most frequent

10000 terms in creating the initialization. This may case the first step of the algorithm to have a very bad value of the objective function but it should quickly stabilize into a good place. You can tweak the exact number where this kicks in with the maxV argument inside control. There appear to be some cases where numerical instability in the Spectral algorithm can cause differences across machines (particularly Windows machines for some reason). It should always give exactly the same answer for a given machine but if you are seeing different answers on different machines, see https://github.com/bstewart/stm/issues/133 for a longer explanation. The other option "LDA" which uses a few passes of a Gibbs sampler is perfectly reproducible across machines as long as the seed is set.

Specifying an integer greater than 1 for the argument ngroups causes the corpus to be broken into the specified number of groups. Global updates are then computed after each group in turn. This approach, called memoized variational inference in Hughes and Sudderth (2013), can lead to more rapid convergence when the number of documents is large. Note that the memory requirements scale linearly with the number of groups so this provides a tradeoff between memory efficiency and speed. The claim of speed here is based on the idea that increasing the number of global updates should help the model find a solution in fewer passes through the document set. However, it is worth noting that for any particular case the model need not converge faster and definitely won't converge to the same location. This functionality should be considered somewhat experimental and we encourage users to let us know what their experiences are like here in practice.

Models can now be restarted by passing an STM object to the argument model. This is particularly useful if you run a model to the maximum iterations and it terminates without converging. Note that all the standard arguments still need to be passed to the object (including any formulas, the number of topics, etc.). Be sure to change the max.em.its argument or it will simply complete one additional iteration and stop.

You can pass a custom initialization of the beta model parameters to stm.

The control argument is a list with named components which can be used to specify numerous additional computational details. Valid components include:

tau.maxit Controls the maximum number of iterations when estimating the prior for content covariates. When the mode is Jeffreys, estimation proceeds by iterating between the kappa vector corresponding to a particular topic and the associated variance tau before moving on to the next parameter vector. this controls the maximum number of iterations. It defaults to NULL effectively enforcing convergence. When the mode is L1 this sets the maximum number of passes in the coordinate descent algorithm and defaults to 1e8.

tau.tol Sets the convergence tolerance in the optimization for content covariates. When the mode is Jeffreys this sets the convergence tolerance in the iteration between the kappa vector and variances tau and defaults to 1e-5. With L1 it defaults to 1e-6.

kappa.mstepmaxit When the mode for content covariate estimation is Jeffreys this controls the maximum number of passes through the sequence of kappa vectors. It defaults to 3. It has no role under L1- see tau.maxit option instead.

kappa.msteptol When the mode for content covariate estimation is Jeffreys this controls the tolerance for convergence (measured by the L1 norm) for the entire M-step. It is set to .01 by default. This has no role under mode L1- see tau.tol option instead.

fixedintercept a logical indicating whether in content covariate models the intercept should be fixed to the background distribution. TRUE by default. This only applies when kappa.prior is set to L1. If FALSE the intercept is estimated from the data without penalty. In practice

estimated intercepts often push term probabilities to zero, resulting in topics that look more like those in a Dirichlet model- that is, most terms have approximately zero probability with some terms with high probability.

`kappa.enet` When using the L1 mode for content covariates this controls the elastic net mixing parameter. See the argument `alpha` in `glmnet`. Value must be between 1 and 0 where 1 is the lasso penalty (the default) and 0 is the ridge penalty. The closer the parameter is to zero the less sparse the solution will tend to be.

`gamma.enet` Controls the elastic net mixing parameter for the prevalence covariates. See above for a description.

`gamma.ic.k` For L1 mode prevalence covariates this controls the selection of the regularization parameter. We use a generic information criterion which penalizes complexity by the parameter `ic.k`. When set to 2 (as by default) this results in AIC. When set to log(n) (where n is the total number of documents in the corpus) this is equivalent to BIC. Larger numbers will express a preference for sparser (simpler) models.

`gamma.maxits` An integer indicating the maximum number of iterations that the prevalence regression variational algorithm can run before erroring out. Defaults to 1000.

`nlambda` Controls the length of the regularization path when using L1 mode for content covariates. Defaults to 500. Note that glmnet relies heavily on warm starts and so a high number will often (counter-intuitively) be less costly than a low number. We have chosen a higher default here than the default in the glmnet package and we don't recommend changing it.

`lambda.min.ratio` For L1 mode content covariates this controls the explored path of regularization values. This defaults to .0001. Setting higher numbers will result in more sparse solutions. This is here primarily for dealing with convergence issues, if you want to favor selection of sparser solutions see the next argument.

`ic.k` For L1 mode content covariates this controls the selection of the regularization parameter. We use a generic information criterion which penalizes complexity by the parameter `ic.k`. When set to 2 (as by default) this results in AIC. When set to log(n) (where n is the total number of words in the corpus) this is equivalent to BIC. Larger numbers will express a preference for sparser (simpler) models.

`nits` Sets the number of iterations for collapsed gibbs sampling in LDA initializations. Defaults to 50

`burnin` Sets the burnin for collapsed gibbs sampling in LDA initializations. Defaults to 25

`alpha` Sets the prevalence hyperparameter in collapsed gibbs sampling in LDA initializations. Defaults to 50/K

`eta` Sets the topic-word hyperparameter in collapsed gibbs sampling in LDA initializations. Defaults to .01

`contrast` A logical indicating whether a standard contrast coding should be used for content covariates. Typically this should remain at the default of FALSE.

`rp.s` Parameter between 0 and 1 controlling the sparsity of random projections for the spectral initialization. Defaults to .05

`rp.p` Dimensionality of the random projections for the spectral initialization. Defaults to 3000.

`rp.d.group.size` Controls the size of blocks considered at a time when computing the random projections for the spectral initialization. Defaults to 2000.

`SpectralRP` A logical which when `TRUE` turns on the experimental random projections spectral initialization.

`maxV` For spectral initializations this will set the maximum number of words to be used in the initialization. It uses the most frequent words first and then they are reintroduced following initialization. This allows spectral to be used with a large V.

`recoverEG` Set to codeTRUE by default. If set to `FALSE` will solve the recovery problem in the Spectral algorithm using a downhill simplex method. See https://github.com/bstewart/stm/issues/133 for more discussion.

`allow.neg.change` A logical indicating whether the algorithm is allowed to declare convergence when the change in the bound has become negative. Defaults to `TRUE`. Set to `FALSE` to keep the algorithm from converging when the bound change is negative. NB: because this is only an approximation to the lower-bound the change can be negative at times. Right now this triggers convergence but the final approximate bound might go higher if you are willing to wait it out. The logic of the default setting is that a negative change in the bound usually means it is barely moving at all.

`custom.beta` If `init.type="Custom"` you can pass your own initialization of the topic-word distributions beta to use as an initialization. Please note that this takes some care to be sure that it is provided in exactly the right format. The number of topics and vocab must match exactly. The vocab must be in the same order. The values must not be pathological (for instance setting the probability of a single word to be 0 under all topics). The beta should be formatted in the same way as the piece of a returned stm model object `stmobj$beta$logbeta`. It should be a list of length the number of levels of the content covariate. Each element of the list is a K by V matrix containing the logged word probability conditional on the topic. If you use this option we recommend that you use `max.em.its=0` with the model initialization set to random, inspect the returned form of `stmobj$beta$logbeta` and ensure that it matches your format.

`tSNE_init.dims` The K=0 spectral setting uses tSNE to create a low-dimensional projection of the vocab co-occurrence matrix. tSNE starts with a PCA projection as an initialization. We actually do the projection outside the tSNE code so we can use a randomized projection approach. We use the 50 dimensional default of the **rtsne** package. That can be changed here.

`tSNE_perplexity` The `Rtsne` function in the **rtsne** package uses a perplexity parameter. This defaults to 30 and can throw an error when too high. `stm` will automatically lower the parameter for you until it works, but it can also be directly set here.

Value

An object of class STM

`mu`	The corpus mean of topic prevalence and coefficients
`sigma`	Covariance matrix
`beta`	List containing the log of the word probabilities for each topic.
`settings`	The settings file. The Seed object will always contain the seed which can be fed as an argument to recover the model.
`vocab`	The vocabulary vector used.
`convergence`	list of convergence elements including the value of the approximate bound on the marginal likelihood at each step.

theta	Number of Documents by Number of Topics matrix of topic proportions.
eta	Matrix of means for the variational distribution of the multivariate normal latent variables used to calculate theta.
invsigma	The inverse of the sigma matrix.
time	The time elapsed in seconds
version	The version number of the package with which the model was estimated.

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.

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.

Roberts, M., Stewart, B., & Tingley, D. (2016). "Navigating the Local Modes of Big Data: The Case of Topic Models. In Data Analytics in Social Science, Government, and Industry." New York: Cambridge University Press.

See Also

prepDocuments labelTopics estimateEffect

Examples

```
#An example using the Gadarian data.  From Raw text to fitted model using
#textProcessor() which leverages the tm Package
temp<-textProcessor(documents=gadarian$open.ended.response,metadata=gadarian)
out <- prepDocuments(temp$documents, temp$vocab, temp$meta)
set.seed(02138)
mod.out <- stm(out$documents, out$vocab, 3,
               prevalence=~treatment + s(pid_rep), data=out$meta)

#The same example using quanteda instead of tm via textProcessor()
#Note this example works with quanteda version 0.9.9-31 and later
require(quanteda)
gadarian_corpus <- corpus(gadarian, text_field = "open.ended.response")
gadarian_dfm <- dfm(gadarian_corpus,
                    remove = stopwords("english"),
                    stem = TRUE)
stm_from_dfm <- stm(gadarian_dfm, K = 3, prevalence = ~treatment + s(pid_rep),
                    data = docvars(gadarian_corpus))

#An example of restarting a model
```

```
mod.out <- stm(out$documents, out$vocab, 3, prevalence=~treatment + s(pid_rep),
               data=out$meta, max.em.its=5)
mod.out2 <- stm(out$documents, out$vocab, 3, prevalence=~treatment + s(pid_rep),
                data=out$meta, model=mod.out, max.em.its=10)
```

`summary.estimateEffect` *Summary for estimateEffect*

Description

Create a summary regression table similar to those produced for `lm`

Usage

```
## S3 method for class 'estimateEffect'
summary(object, topics = NULL, nsim = 500, ...)
```

Arguments

`object`	an object of class "`estimateEffect`", usually a result of a call to `estimateEffect`
`topics`	a vector containing the topic numbers for each a summary is to be calculated. Must be contained in the original `estimateEffect` object
`nsim`	the number of simulations to use per parameter set to calculate the standard error. Defaults to 500
`...`	further arguments passed to or from other methods

Details

This function along with `print.summary.estimateEffect` creates regression tables that look like typically summaries you see in R. In general we recommend that you use non-linearities such as splines via function like `s` and in those circumstances the tables are not particularly interpretable.

Confidence intervals are calculated by using draws from the covariance matrix of each simulation to estimate the standard error. Then a t-distribution approximation is applied to calculate the various quantities of interest.

See Also

`estimateEffect` `plot.estimateEffect`

summary.STM *Summary Function for the STM objects*

Description

Function to report on the contents of STM objects

Usage

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

Arguments

object	An STM object.
...	Additional arguments affecting the summary

Details

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

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 **tm** package where various user specified options can be selected.

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 insures 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 SMART stopword list (in English) or the snowball stopword lists (for all other languages). 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) Note that languages listed here may not all have accompanying stopwords. However if you have your own stopword list you can use customstopwords below.
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. Defaults to FALSE. Runs before call to tm package.

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

`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 standard 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 see also the `ucp` argument to turn on a stronger punctuation filter). This can in theory be used to remove any characters you don't want in the text for some reason. In practice what this function does is collapse the character vector to one string and put square brackets around it in order to make a pattern that can be matched and replaced with `gsub` at the punctuation removal stage. If the `custompunctuation` vector is length 1 and the first element is a left square bracket, the function assumes that you have passed a regular expression and passes that directly along to `gsub`.

`v1` A logical which defaults to `FALSE`. If set to `TRUE` it will use the ordering of operations we used in Version 1.0 of the package.

Details

This function is designed to provide a convenient and quick way to process a relatively small volume 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. If you have texts more complicated than a spreadsheet, we recommend you check out the excellent **readtext** package.

The processor always strips extra white space but all other processing options are optional. Stemming uses the snowball stemmers and supports a wide variety of languages. Words in the vocabulary can be dropped due to sparsity and stop word removal. If a document no longer contains any words it is dropped from the output. Specifying meta-data is a convenient way to make sure the appropriate rows are dropped from the corresponding metadata file.

When the option `sparseLevel` is set to a number other than 1, infrequently appearing words are removed. When a term is removed from the vocabulary a message will print to the screen (as long as `verbose` has not been set to `FALSE`). The message indicates the number of terms removed (that is, the number of vocabulary entries) as well as the number of tokens removed (appearances of individual words). The function `prepDocuments` provides additional methods to prune infrequent words. In general the functionality there should be preferred.

We emphasize that this function is a convenience wrapper around the excellent **tm** package functionality without which it wouldn't be possible.

Value

documents	A list containing the documents in the stm format.
vocab	Character vector of vocabulary.
meta	Data frame or matrix containing the user-supplied metadata for the retained documents.

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.

See Also

readCorpus

Examples

```
head(gadarian)
#Process the data for analysis.
temp<-textProcessor(documents=gadarian$open.ended.response,metadata=gadarian)
meta<-temp$meta
vocab<-temp$vocab
docs<-temp$documents
out <- prepDocuments(docs, vocab, meta)
docs<-out$documents
vocab<-out$vocab
meta <-out$meta


#Example of custom punctuation removal.
docs <- c("co.rr?ec!t")
textProcessor(docs,custompunctuation=c(".","?","!"),
              removepunctuation = FALSE)$vocab
#note that the above should now say "correct"
```

---

thetaPosterior *Draw from Theta Posterior*

Description

Take random draws from the variational posterior for the document-topic proportions. This is underlying methodology for estimateEffect

Usage

```
thetaPosterior(
  model,
  nsims = 100,
  type = c("Global", "Local"),
  documents = NULL
)
```

Arguments

model	An STM object created by stm
nsims	The number of draws from the variational posterior. See details below.
type	A choice of two methods for constructing the covariance approximation the "Global" approximation and the "Local" approximation. See details below.
documents	If type="Local", the documents object used in the original stm call should be passed here.

Details

This function allows the user to draw samples from the variational posterior distribution over the normalized document-topic proportions, theta. The function estimateEffect provides a user-friendly interface for running regressions using samples from the posterior distribution. When the user wants to do something not covered by that function, the code here provides easy access to uncertainty in the model.

In order to simulate from the variational posterior for theta we take draws from the variational distribution for eta (the unnormalized topic proportions) and then map them to the simplex. Each document in the corpus has its own mean vector (eta) and covariance matrix (nu). Because the covariance matrices can be large we do not store them in the model objects. We offer two approximations to the covariance matrix: Global and Local. The Global method constructs a single approximate covariance matrix which is then shared by all documents. This approach is very fast and does not require access to the original documents. For highly aggregated quantities of interest this often produces similar results to the Local method.

The Local method steps through each document in sequence and calculates the covariance matrix. If the model has not converged, this matrix can be undefined and so we perform document level inference until the estimate stabilizes. This means that under the Local method both the covariance and the mean of the variational distribution are recalculated. It also means that calling this option with Local specified will take approximately as long as a standard E-step of stm for the same data and possibly longer. Because the memory requirements would be extreme for large K, we calculate one document at a time, discarding the covariance matrix before proceeding to the next document. Thus, if your computer has sufficient memory it is dramatically more computationally efficient to draw all the samples you may require at once rather than taking one sample at a time.

The output for both methods is a list with number of elements equal to the number of documents. Each element is a matrix with nsims rows and K columns. Be careful to ensure that you have sufficient memory before attempting this with a large number of simulations, documents or topics.

See Also

estimateEffect

Examples

```
#global approximation
draws <- thetaPosterior(gadarianFit, nsims = 100)
```

---

toLDAvis *Wrapper to launch LDAvis topic browser.*

Description

Tool for exploring topic/word distributions using LDAvis topic browser.

Usage

```
toLDAvis(
  mod,
  docs,
  R = 30,
  plot.opts = list(xlab = "PC1", ylab = "PC2"),
  lambda.step = 0.1,
  out.dir = tempfile(),
  open.browser = interactive(),
  as.gist = FALSE,
  reorder.topics = TRUE
)
```

Arguments

mod	STM object. Output from stm function.
docs	Documents object passed to stm in this package's standard format (see the documentation in stm.
R	Passed to createJSON "integer, the number of terms to display in the barcharts of the interactive viz. Default is 30. Recommended to be roughly between 10 and 50."
plot.opts	Passed to createJSON "a named list used to customize various plot elements. By default, the x and y axes are labeled 'PC1' and 'PC2' (principal components 1 and 2), since jsPCA is the default scaling method. "
lambda.step	Passed to createJSON "a value between 0 and 1. Determines the interstep distance in the grid of lambda values over which to iterate when computing relevance. Default is 0.01. Recommended to be between 0.01 and 0.1."
out.dir	Passed to serVis "directory to store html/js/json files."

open.browser	Passed to serVis "Should R open a browser? If yes, this function will attempt to create a local file server via the servr package. This is necessary since the javascript needs to access local files and most browsers will not allow this."
as.gist	Passed to serVis "should the vis be uploaded as a gist? Will prompt for an interactive login if the GITHUB_PAT environment variable is not set"
reorder.topics	Passed to createJSON "Should LDAvis re-order the K topics in order of decreasing proportion? Default is True"

Details

Tool for exploring topic/word distributions using LDAvis topic browser. Development build of LDAvis available at https://github.com/cpsievert/LDAvis or download from CRAN. Note: LDAvis may renumber the topics.

References

Carson Sievert and Kenny Shirley. LDAvis: Interactive Visualization of Topic Models. R package version 0.3.1. https://github.com/cpsievert/LDAvis

Examples

```
mod <- stm(poliblog5k.docs, poliblog5k.voc, K=25,
           prevalence=~rating, data=poliblog5k.meta,
           max.em.its=2, init.type="Spectral")
#please don't run a model with 2 iterations
#this is done here to make it run quickly.
toLDAvis(mod=mod, docs=poliblog5k.docs)
```

---

toLDAvisJson *Wrapper to create Json mapping for LDAvis. This can be useful in indirect render e.g. Shiny Dashboards*

Description

Tool for exploring topic/word distributions using LDAvis topic browser.

Usage

```
toLDAvisJson(
  mod,
  docs,
  R = 30,
  plot.opts = list(xlab = "PC1", ylab = "PC2"),
```

```
  lambda.step = 0.1,
  reorder.topics = TRUE
)
```

Arguments

mod	STM object. Output from stm function.
docs	Documents object passed to stm in this package's standard format (see the documentation in stm.
R	Passed to createJSON "integer, the number of terms to display in the barcharts of the interactive viz. Default is 30. Recommended to be roughly between 10 and 50."
plot.opts	Passed to createJSON "a named list used to customize various plot elements. By default, the x and y axes are labeled 'PC1' and 'PC2' (principal components 1 and 2), since jsPCA is the default scaling method. "
lambda.step	Passed to createJSON "a value between 0 and 1. Determines the interstep distance in the grid of lambda values over which to iterate when computing relevance. Default is 0.01. Recommended to be between 0.01 and 0.1."
reorder.topics	Passed to createJSON "Should LDAvis re-order the K topics in order of decreasing proportion? Default is True"

Details

Tool for exploring topic/word distributions using LDAvis topic browser. Development build of LDAvis available at https://github.com/cpsievert/LDAvis or download from CRAN. Note: LDAvis may renumber the topics.

References

Carson Sievert and Kenny Shirley. LDAvis: Interactive Visualization of Topic Models. R package version 0.3.1. https://github.com/cpsievert/LDAvis

Examples

```
mod <- stm(poliblog5k.docs, poliblog5k.voc, K=25,
           prevalence=~rating, data=poliblog5k.meta,
           max.em.its=2, init.type="Spectral")
#please don't run a model with 2 iterations
#this is done here to make it run quickly.
toLDAvisJson(mod=mod, docs=poliblog5k.docs)
```

`topicCorr` *Estimate topic correlation*

Description

Estimates a graph of topic correlations using either a simple thresholding measure or more sophisticated tests from the package huge.

Usage

```
topicCorr(model, method = c("simple", "huge"), cutoff = 0.01, verbose = TRUE)
```

Arguments

`model`	An STM object for which you want to estimate correlations between topics.
`method`	Method for estimating the graph. `"simple"` simply thresholds the covariances. `"huge"` uses the semiparametric procedure in the package huge. See details below.
`cutoff`	When using the simple method, this is the cutoff below which correlations are truncated to zero.
`verbose`	A logical which indicates whether information should be printed to the screen when running `"huge"`.

Details

We offer two estimation procedures for producing correlation graphs. The results of either method can be plotted using `plot.topicCorr`. The first method is conceptually simpler and involves a simple thresholding procedure on the estimated marginal topic proportion correlation matrix and requires a human specified threshold. The second method draws on recent literature undirected graphical model estimation and is automatically tuned.

The `"simple"` method calculates the correlation of the MAP estimates for the topic proportions θ which yields the marginal correlation of the mode of the variational distribution. Then we simply set to 0 those edges where the correlation falls below the threshold.

An alternative strategy is to treat the problem as the recovery of edges in a high-dimensional undirected graphical model. In these settings we assume that observations come from a multivariate normal distribution with a sparse precision matrix. The goal is to infer which elements of the precision matrix are non-zero corresponding to edges in a graph. Meinshausen and Buhlmann (2006) showed that using sparse regression methods like the LASSO it is possible to consistently identify edges even in very high dimensional settings.

Selecting the option `"huge"` uses the huge package by Zhao and Liu to estimate the graph. We use a nonparanormal transformation of the topic proportions (θ) which uses semiparametric Gaussian copulas to marginally transform the data. This weakens the gaussian assumption of the subsequent procedure. We then estimate the graph using the Meinshausen and Buhlmann procedure. Model selection for the scale of the L_1 penalty is performed using the rotation information criterion (RIC) which estimates the optimal degree of regularization by random rotations. Zhao and Lieu (2012)

note that this selection approach has strong empirical performance but is sensitive to under-selection of edges. We choose this metric as the default approach to model selection to reflect social scientists' historically greater concern for false positive rates as opposed to false negative rates.

We note that in models with low numbers of topics the simple procedure and the more complex procedure will often yield identical results. However, the advantage of the more complex procedure is that it scales gracefully to models with hundreds or even thousands of topics - specifically the set of cases where some higher level structure like a correlation graph would be the most useful.

Value

`posadj`	K by K adjacency matrix where an edge represents positive correlation selected by the model.
`poscor`	K by K correlation matrix. It takes values of zero where the correlation is either negative or the edge is unselected by the model selection procedure.
`cor`	K by K correlation matrix element-wise multiplied by the adjacency matrix. Note that this will contain significant negative correlations as well as positive correlations.

References

Lucas, Christopher, Richard A. Nielsen, Margaret E. Roberts, Brandon M. Stewart, Alex Storer, and Dustin Tingley. "Computer-Assisted Text Analysis for Comparative Politics." Political Analysis (2015).

T. Zhao and H. Liu. The huge Package for High-dimensional Undirected Graph Estimation in R. Journal of Machine Learning Research, 2012

H. Liu, F. Han, M. Yuan, J. Lafferty and L. Wasserman. High Dimensional Semiparametric Gaussian Copula Graphical Models. Annals of Statistics,2012

N. Meinshausen and P. Buhlmann. High-dimensional Graphs and Variable Selection with the Lasso. The Annals of Statistics, 2006.

See Also

`plot.topicCorr`

`topicLasso` *Plot predictions using topics*

Description

Use the **glmnet** package to plot LASSO based estimates of relationship between an arbitrary dependent variable with topics and additional variables as predictors. This function is experimental (see below).

Usage

```
topicLasso(
  formula,
  data,
  stmobj = NULL,
  subset = NULL,
  omit.var = NULL,
  family = "gaussian",
  main = "Topic Effects on Outcome",
  xlab = expression("Lower Outcome Higher Outcome"),
  labeltype = c("prob", "frex", "lift", "score"),
  seed = 2138,
  xlim = c(-4, 4),
  standardize = FALSE,
  nfolds = 20,
  ...
)
```

Arguments

formula	Formula specifying the dependent variable and additional variables to included in the LASSO beyond the topics present in the stmobj. Just pass a 1 on the right-hand side in order to run without additional controls.
data	Data file containing the dependent variable. Typically will be the metadata file used in the stm analysis. It must have a number of rows equal to the number of documents in the stmobj.
stmobj	The STM object, and output from the stm function.
subset	A logical statement that will be used to subset the corpus.
omit.var	Pass a character vector of variable names to be excluded from the plot. Note this does not exclude them from the calculation, only the plot.
family	The family parameter used in glmnet. See explanation there. Defaults to "gaussian"
main	Character string for the main title.
xlab	Character string giving an x-axis label.
labeltype	Type of example words to use in labeling each topic. See labelTopics. Defaults to "prob".
seed	The random seed for replication of the cross-validation samples.
xlim	Width of the x-axis.
standardize	Whether to standardize variables. Default is FALSE, which is different from the **glmnet** default because the topics are already standardized. Note that glmnet standardizes the variables by default but then projects them back to their original scales before reporting coefficients.
nfolds	the number of cross-validation folds. Defaults to 20.
...	Additional arguments to be passed to glmnet. This can be useful for addressing convergence problems.

Details

This function is used for estimating the most important topical predictors of an arbitrary outcome. The idea is to run an L1 regularized regression using `cv.glmnet` in the **glmnet** package where the document-level dependent variable is chosen by the user and the predictors are the document-topic proportions in the `stm` model along with any other variables of interest.

The function uses cross-validation to choose the regularization parameter and generates a plot of which loadings were the most influential in predicting the outcome. It also invisibly returns the glmnet model so that it can be used for prediction.

NOTE: This function is still very experimental and may have stability issues. If stability issues are encountered see the documentation in **glmnet** for arguments that can be passed to improve convergence. Also, it is unlikely to work well with multivariate gaussian or multinomial families.

References

Friedman, Jerome, Trevor Hastie, and Rob Tibshirani. "Regularization paths for generalized linear models via coordinate descent." Journal of statistical software 33.1 (2010): 1.

See Also

glmnet

Examples

```
#Load the poliblog data
data(poliblog5k)
#estimate a model with 50 topics
stm1 <- stm(poliblog5k.docs, poliblog5k.voc, 50,
            prevalence=~rating + blog, data=poliblog5k.meta,
            init.type="Spectral")
#make a plot of the topics most predictive of "rating"
out <- topicLasso(rating ~ 1, family="binomial", data=poliblog5k.meta,stmobj=stm1)
#generate some in-sample predictions
pred <- predict(out, newx=stm1$theta,type="class")
#check the accuracy of the predictions
table(pred, poliblog5k.meta$rating)
```

---

topicQuality *Plots semantic coherence and exclusivity for each topic.*

Description

Plots semantic coherence and exclusivity for each topic. Does not support models with content covariates.

Usage

```
topicQuality(
  model,
  documents,
  xlab = "Semantic Coherence",
  ylab = "Exclusivity",
  labels = 1:ncol(model$theta),
  M = 10,
  ...
)
```

Arguments

model	Output from stm, or a selected model from selectModel.
documents	The documents (see stm for format).
xlab	Character string that is x axis title. This should be semantic coherence.
ylab	Character string that is y axis title. This should be exclusivity.
labels	Vector of number corresponding to topic numbers.
M	Number of words to use in semantic coherence and exclusivity calculations
...	Other plotting parameters from igraph.

Details

Each model has semantic coherence and exclusivity values associated with each topic. This function plots these values and labels each with its topic number.

Examples

```
## Not run:

  #Semantic Coherence calculations require the original documents so we need
  #to reconstruct them here.
  temp<-textProcessor(documents=gadarian$open.ended.response,metadata=gadarian)
  meta<-temp$meta
  vocab<-temp$vocab
  docs<-temp$documents
  out <- prepDocuments(docs, vocab, meta)
  docs<-out$documents
  vocab<-out$vocab
  meta <-out$meta
  topicQuality(model=gadarianFit, documents=docs)

## End(Not run)
```

writeLdac *Write a .ldac file*

Description

A function for writing documents out to a .ldac formatted file.

Usage

```
writeLdac(documents, file, zeroindex = TRUE)
```

Arguments

documents	A documents object to be written out to a file. 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
file	A character string giving the name of the file to be written. This object is passed directly to the argument con in writeLines and thus can be a connection object as well.
zeroindex	If TRUE (the default) it subtracts one from each index. If FALSE it uses the indices as given. The standard .ldac format indexes from 0 as per standard convention in most languages. Our documents format indexes from 1 to abide by conventions in R. This option converts to the zero index by default.

Details

This is a simple convenience function for writing out document corpora. Files can be read back into R using readCorpus or simply used for other programs. The output is a file in the .ldac sparse matrix format popularized by Dave Blei's C code for LDA.

See Also

readCorpus

Examples

```
## Not run:

#Convert the gadarian data into documents format
temp<-textProcessor(documents=gadarian$open.ended.response,metadata=gadarian)
documents <- temp$documents
#Now write out to an ldac file
writeLdac(documents, file="gadarian.ldac")

## End(Not run)
```

Index

∗ **datasets**
gadarian, 20
poliblog5k, 51
∗ **multimodality**
multiSTM, 26
plot.MultimodDiagnostic, 38
∗ **package**
stm-package, 3
∗ **stm**
multiSTM, 26
plot.MultimodDiagnostic, 38

alignCorpus, 4, *17*, *19*
asSTMCorpus, 5

bs, *56*, *63*

calcfrex, *21, 22*
calclift, *21, 22*
calcscore, *21, 22*
checkBeta, 6
checkResiduals, 7, *58*
cloud, 8
convert, *63*
convertCorpus, 10
createJSON, *76–78*
cv.glmnet, *82*

dfm, *5, 6*, *63*
dfm_select, *17*

estimateEffect, *3*, 11, *12*, *29*, *33, 34*, *50*, *65*, *69, 70*, *74–76*
eval.heldout, *58*
eval.heldout *(*make.heldout*)*, 23
exclusivity, *58*

findThoughts, *3*, 13, *16*, *22*, *47, 48*
findTopic, 15
fitNewDocuments, *4, 5*, 16

gadarian, *3*, 20
gadarianFit, *3*
gadarianFit *(*gadarian*)*, 20
glmnet, *81*

head.textProcessor *(*textProcessor*)*, 71

js.estimate, *21, 22*

labelTopics, *3*, 21, *42*, *69*, *71*, *81*
lda, *21*, *63*
loess, *50*

make.dt, *14*, 22
make.heldout, *19*, 23, *58, 59*
makeDesignMatrix, *18, 19*
manyTopics, *3*, 24
multiSTM, 26, *38*, *40*

ns, *56*, *63*

optim, *31*
optimizeDocument, *18, 19*, 30

par, *42*
permutationTest, *3*, 32, *44*
plot.estimateEffect, *3*, *13*, 34, *50*, *70*
plot.findThoughts *(*findThoughts*)*, 13
plot.MultimodDiagnostic, *28, 29*, 38
plot.searchK, 40, *59*
plot.STM, *3*, *9*, *22*, 41
plot.STMpermute, *3*, *33*, 44
plot.topicCorr, *3*, *43*, 45, *79, 80*
plotModels, *3*, 46
plotQuote, *3*, *15*, *43*, 47
plotRemoved, 48, *53*
plotTopicLoess, *3*, 49
poliblog5k, *3*, *10*, 51
predict, *56*
prepDocuments, *3*, *5, 6*, *24, 25*, *49*, 52, *54, 55*, *60*, *63*, *69*, *73*

print.findThoughts *(findThoughts)*, 13
print.labelTopics *(labelTopics)*, 21
print.MultimodDiagnostic *(multiSTM)*, 26
print.sageLabels *(sageLabels)*, 56
print.STM *(summary.STM)*, 71
print.summary.estimateEffect, *70*
print.summary.estimateEffect
(summary.estimateEffect), 70
print.textProcessor *(textProcessor)*, 71

readCorpus, *3*, *10*, 54, *55*, *74*, *84*
readLdac, *54*, 55

s, 56, *63*, *65*, *70*
sageLabels, *18*, 56
searchK, *3*, 57
selectModel, *3*, *29*, 59
semanticCoherence, *58*
serVis, *76, 77*
simple_triplet_matrix, *54*
stm, *3–7*, *9–11*, *18*, *22*, *23*, *25*, *31*, *33*, *52*, *58*, 62, *75, 76*, *78*, *82, 83*
stm-package, 3
summary.estimateEffect, *13*, 70
summary.STM, *3*, 71
summary.textProcessor *(textProcessor)*, 71

textProcessor, *3*, *54, 55*, *63*, 71
thetaPosterior, *32*, 74
toLDAvis, 76
toLDAvisJson, 77
topicCorr, *3*, *45, 46*, 79
topicLasso, 80
topicQuality, *3*, 82

writeLdac, *10*, 84
writeLines, *84*