Package 'stm'

February 28, 2014

Type Package

Title Estimation of the Structural Topic Model

Version 0.6.1

Date 2014-02-28

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

Maintainer Brandon Stewart <bstewart@fas.harvard.edu>

Depends R (>= 2.10)

Imports matrixStats, splines, slam, lda, stringr

Suggests igraph, SnowballC, tm, huge, glmnet, Matrix, textir

LazyData yes

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.

License MIT + file LICENSE

URL http://structuraltopicmodel.com

NeedsCompilation no

Repository CRAN

Date/Publication 2014-02-28 17:55:17

R topics documented:

`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.

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

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

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

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

Plotting Functions `plot.STM` `plot.estimateEffect` `plot.topicCorr` `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 <bstewart@fas.harvard.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. http://scholar.harvard.edu/files/bstewart/files/stmnips2013.pdf

Roberts, M., Stewart, B., Tingley, D., Lucas, C., Leder-Luis, J., Gadarian, S., Albertson, B., Albertson, B. and Rand, D. (Forthcoming). "Structural topic models for open ended survey responses." American Journal of Political Science http://scholar.harvard.edu/files/dtingley/files/topicmodelsopenendedexperiments.pdf

See Also

stm

`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 = 100)
```

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 100.

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.

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.

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.

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`

Examples

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

```
#three topics at once
## Not run:
prep <- estimateEffect(1:3 ~ treatment, gadarianFit, gadarian)
plot.estimateEffect(prep, "treatment", model=gadarianFit, method="pointestimate")

## End(Not run)
#See vignette for examples of ploting models with an interaction.
```

`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)
```

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.
`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.

Details

Prints to the screen 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). Returns document indices and top thoughts.

Value

`index`	Matrix with one column per requested topic and one row for each requested document. Entries are the document indices within the dataset.
`docs`	List of top thoughts in the order of topics.

Examples

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

gadarian *Gadarian and Albertson data*

Description

This data set contains variables from Gadarian and Albertson, forthcoming, "Anxiety, Immigration, and the Search for Information", Political Psychology. 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.

Usage

```
data(gadarian)
```

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 and Albertson, forthcoming, "Anxiety, Immigration, and the Search for Information", Political Psychology http://onlinelibrary.wiley.com/doi/10.1111/pops.12034/abstract

Roberts, M., Stewart, B., Tingley, D., Lucas, C., Leder-Luis, J., Gadarian, S., Albertson, B., Albertson, B. and Rand, D. (Forthcoming). "Structural topic models for open ended survey responses." American Journal of Political Science http://scholar.harvard.edu/files/dtingley/files/topicmodelsopenendedexperiments.pdf

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.
`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 $p(z|w = v)$.

Score and Lift are measures provided in two other popular text mining packages. For more information on type Score, see the R package `lda`. For more information on type Lift, see Taddy, "Multinomial Inverse Regression for Text Analysis", Journal of the American Statistical Association 108, 2013 and the R package `textir`.

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

Examples

```
labelTopics(gadarianFit)
```

---

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

Description

Works the same as modelSelect, 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, content, data=NULL,
                        max.em.its=100, verbose=TRUE, init.type =
                        "LDA",
                        emtol= 1e-05, seed=NULL,runs=50, frexw=.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 w\ ith 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 sec\ ond 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 smoo\ th 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 fo\ r each document.
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.
data	Dataset which contains prevalence and content covariates.
init.type	The method of initialization. Must be either Latent Dirichlet Allocation (LDA), Dirichlet Multinomial Regression Topic Model (DMR), a random initialization (Random)\ or a previous STM object (User).
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 repro\ duce 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.
emtol	Convergence tolerance. EM stops when the relative change in the approximate bound drops below this level. Defaults to .001%.
verbose	A logical flag indicating whether information should be printed to the screen.
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

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 selected for each number of topics.
exclusivity	Exclusivity values for each topic wihtin 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)
```

---

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="pointestimate",
                    cov.value1=NULL, cov.value2=NULL, int.value=NULL,
                    npoints=100, nsims=100, ci.level=.95,
                    xlim=NULL, ylim=NULL, ylab="",
                    main="",printlegend=TRUE, labeltype="numbers",
                    n=7,frexw=.5, xlab="",
                    add=FALSE, linecol=NULL, width=25, verbose.labels=TRUE,
                    ...)
```

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".
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.
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 input a vector for labeltype with their own labels. Labels appear in the legend for continous covariates.
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.
int.value	For methods "pointestimate" and "difference", when two binary 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.
n	Number of words to print if "prob", "score", "lift", or "frex" is chosen.
ci.level	Confidence level for confidence intervals.
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.
verbose.labels	For method "difference" – verboselabels will specify the comparison covariate values of the covariate on the plot.
xlim	Vector of x axis minimum and maximum values.
ylim	Vector of y axis minimum and maximum values.
main	Character string that is plot title.
xlab	Character string that is x axis title.

ylab	Character string that is y axis title.
width	Character string that is y axis title.
...	Other plotting parameters

Examples

```
## Not run:
prep <- estimateEffect(1:3 ~ treatment, gadarianFit, gadarian)
plot.estimateEffect(prep, "treatment", model=gadarianFit,
method="pointestimate")
plot.estimateEffect(prep, "treatment", model=gadarianFit,
method="pointestimate")
plot.estimateEffect(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.

#Example of binary times binary interaction
gadarian$binaryvar <- sample(c(0,1), nrow(gadarian), replace=T)
temp <- textProcessor(gadarian$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.estimateEffect(prep, "binaryvar", method="pointestimate",
cov.value1=1, cov.value2=0, xlim=c(-1,1), int.value=1)
plot.estimateEffect(prep, "binaryvar", method="pointestimate",
cov.value1=1, cov.value2=0, xlim=c(-1,1), int.value=0)

## End(Not run)
```

---

plot.STM *Plot summary of an STM object*

Description

Produces one of three 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.

Usage

```
## S3 method for class 'STM'
plot(x, type = c("summary", "labels", "perspectives"),
                  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, ...)
```

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.
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
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.
...	Additional parameters passed to ploting 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.

References

Roberts, M., Stewart, B., Tingley, D., Lucas, C., Leder-Luis, J., Gadarian, S., Albertson, B., Albertson, B. and Rand, D. (Forthcoming). "Structural topic models for open ended survey responses." American Journal of Political Science http://scholar.harvard.edu/files/dtingley/files/topicmodelsopenendedexperiments.pdf

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.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 = layout.fruchterman.reingold,
              vertex.color = "green", vertex.label.cex = 0.75,
              vertex.label.color = "black", ...)
```

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.
vertex.color	Color of the vertices.
vertex.label.cex	Controls the size of the labels.
vertex.label.color	Controls the color of the labels.
...	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),...)
```

Arguments

models	output from selectModel.
labels	labels for each model.
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.
...	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, main=NULL)
```

Arguments

`sentences`	Vector of sentence to plot.
`width`	Number of characters in each line.
`main`	Title of plot.

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)
```

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=.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

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.

Usage

```
poliblog5k.meta
```

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)
## Not run:
stm1 <- stm(poliblog5k.docs, poliblog5k.voc, 3,
prevalence=~rating, data=poliblog5k.meta)

## End(Not run)
```

`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 unusued words in the vocab vector).

Usage

```
prepDocuments(documents, vocab, meta,
              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 ot print details to the screen.

Details

The default setting `dropthresh=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.

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.
`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.

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
```

---

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", "ldac", "slam", "Matrix", "txtorgvocab"))
```

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 ldac takes a file path and reads in a document in the sparse format popularized by David Blei's C code implementation of lda. 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.

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.

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.

Finally the object txtorgvocab allows the user to easily read in a vocab file generated by the software txtorg. When working in English it is straightforward to read in files created by txtorg. However when working in other languages, particularly Chinese and Arabic, there can often be difficulty reading in the files using read.table or read.csv This function should work well in those circumstances.

Value

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

See Also

textProcessor, prepDocuments

Examples

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

---

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.

Value

a predictor matrix of the basis functions.

See Also

bs ns

`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, content, data=NULL,
                        max.em.its=100, verbose=TRUE, init.type =
                        "LDA",
                        emtol= 1e-05, seed=NULL,runs=50, frexw=.7,
                        net.max.em.its=2, netverbose=FALSE, M=10,N=NULL,...)
```

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 w\ ith 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 sec\ ond 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 us\ ing 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 smoo\ th 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 fo\ r each document.
`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.
`data`	Dataset which contains prevalence and content covariates.

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.
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.
max.em.its	The maximum number of EM iterations. If convergence has not been met at this point, a message will be printed.
emtol	Convergence tolerance. EM stops when the relative change in the approximate bound drops below this level. Defaults to .001%.
verbose	A logical flag indicating whether information should be printed to the screen.
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.
...	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 wihtin 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)

## 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 documents, 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, content, data=NULL,
    init.type=c("LDA", "DMR","Random"), seed=NULL,
    max.em.its=100, emtol=1e-5,
    verbose=TRUE, reportevery=5, keepHistory=FALSE,
    LDAbeta=TRUE, interactions=TRUE,
    gamma.prior=c("Pooled", "L1"), sigma.prior=0,
    kappa.prior=c("Jeffreys", "L1"), control=list())
```

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`. Raw texts can be ingested using `textProcessor`.
`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`	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. Must be either Latent Dirichlet Allocation (LDA), Dirichlet Multinomial Regression Topic Model (DMR), or a random initialization. If you want to replicate a previous result, see the argument `seed`.
`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.
`emtol`	Convergence tolerance. EM stops when the relative change in the approximate bound drops below this level. Defaults to .001.
`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.
`keepHistory`	Logical indicating whether the history should be saved at each iteration. Defaults to `FALSE`. Note that the model parameters are extremely memory intensive so use with care.
`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.
`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 broad gamma hyperprior. The alternative `L1` uses `glmnet` to estimate a grouped penalty between L1-L2. 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 is `Jeffreys` and uses a scale mixture of Normals with an improper Jeffreys prior. The option `L1` uses `glmnet` to estimate with a penalty between L1 and L2, See details for more.
`control`	a list of additional parameters control portions of the optimization. See details.

Details

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 unidimensional scaling model. For short corpora focused on very specific subject matter (such as survey experiments) 3-5 topics is a useful starting range. For small corpora (a few hundred to a few thousand) 5-20 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 50-60 topics to work well. For larger corpora 100 topics is a useful default size. Of course, your mileage may vary.

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.

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 and content, 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` and `kappa.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 `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` as by default, 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 .99 by default. This has no role under mode `L1`- see `tau.tol` option instead.

`wordconverge.num` provides an alternative convergence metric based on the number of iterations without change in the top n most probable words per topic. This sets the number n. This is 20 by default (although it is deactivated by default as well- see the next argument).

`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.

`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.

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.
history	If keepHistory=TRUE the history of model parameters at each step.

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. http://scholar.harvard.edu/files/bstewart/files/stmnips2013.pdf

Roberts M., Stewart, B. and Airoldi, E. (2014) "Structural Topic Models"

Roberts, M., Stewart, B., Tingley, D., Lucas, C., Leder-Luis, J., Gadarian, S., Albertson, B., Albertson, B. and Rand, D. (Forthcoming). "Structural topic models for open ended survey responses." American Journal of Political Science http://scholar.harvard.edu/files/dtingley/files/topicmodelsopenendedexperiments.pdf

See Also

prepDocuments labelTopics estimateEffect

Examples

```
## Not run:
#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)
mod.out <- stm(docs, vocab, 3, prevalence=~treatment + s(pid_rep), data=meta)



## End(Not run)
```

---

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, stem=TRUE,
              sparselevel=.99, language="en",
              verbose=TRUE)
```

Arguments

documents	The documents to be processed. A character vector where each entry is the full text of a document. The tm package has a variety of extra readers for ingesting other file formats (.doc, .pdf, .txt, .xml).
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.
stem	Whether or not to stem words. Defaults to TRUE
sparselevel	removes terms where at least sparselevel proportion of the entries are 0.

language	Language used for processing. Defaults to English. tm uses the SnowballC stemmer which as of version 0.5 supports "danish dutch english finnish french german hungarian italian norwegian portuguese romanian russian spanish swedish turkish". These can be specified as any on of the above strings or by the three-letter ISO-639 codes. You can also set language to "na" if you want to leave it deliberately unspecified (see documentation in tm)
verbose	If true prints information as it processes.

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. Once the text has been processed by tm the document term matrix is converted to the stm format using readCorpus.

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.

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
```

---

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. Meinhuasen 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 Meinhuasen and Buhlman 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 C, Nielsen R, Roberts ME, Stewart BM, Storer A, Tingley D. Computer assisted text analysis for comparative politics.

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`

`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	Vector containing documents used.
labels	Vector of number corresponding to topic numbers.
M	Number of words to use in semantic coherence and exclusivity calculations
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.
...	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)
```

Index