Package: faSTM (via r-universe)

June 27, 2026

Title Fast Structural Topic Models

Version 0.0.0.9000

Description A modern implementation of the Structural Topic Model. faSTM fits the logistic-normal STM (with prevalence and content covariates) via a multithreaded Rust core, with an opt-in stochastic-variational path for large corpora. It is self-contained: text preparation is read from 'quanteda' or 'tidytext' objects, model inspection (labelTopics with FREX/lift/score, findThoughts, semantic coherence, exclusivity, topic correlations) and an estimateEffect() (method-of-composition posterior propagation) are built in. The fitted object is structurally compatible with 'stm' so existing analyses migrate with minimal changes.

License Apache License (>= 2)

Encoding UTF-8

Roxygen list(markdown = TRUE)

RoxygenNote 7.3.2

SystemRequirements Cargo (Rust's package manager), rustc

Imports MASS, generics, Matrix, methods, parallel, splines, stats

Suggests ggplot2, ggrepel, glmnet, huge, igraph, LDAvis, lme4, clue, quanteda, stm, tidytext, knitr, rmarkdown, testthat (>= 3.0.0)

Config/rextendr/version 0.3.1

Config/testthat/edition 3

URL https://nealcaren.github.io/faSTM/, https://github.com/nealcaren/faSTM

BugReports https://github.com/nealcaren/faSTM/issues

LazyData true

VignetteBuilder knitr

Config/pak/sysreqs libclang-dev

Repository https://nealcaren.r-universe.dev

Date/Publication 2026-06-27 16:12:57 UTC

RemoteUrl https://github.com/nealcaren/faSTM

RemoteRef HEAD

RemoteSha b4ac5c1f0d3ec094b2d5fe06ce9adb6edd9ff34a

Contents

`align_corpus` *Align a new corpus to a fitted model's vocabulary*

Description

Maps a new corpus's terms onto the term indices of a fitted faSTM model, dropping out-of-vocabulary terms — the preprocessing needed before inferring topics for new documents (cf. `stm::alignCorpus`).

Usage

```
align_corpus(newdata, model)
```

Arguments

`newdata`	A `faSTM_corpus`, quanteda `dfm`, or document-term matrix.
`model`	A faSTM fit.

Value

A list with per-document `ids` (0-based indices into `model$vocab`) and `counts`, plus `dropped` (count of out-of-vocabulary term tokens).

alignCorpus — *Align a new corpus to a reference vocabulary (stm-compatible)*

Description

stm-shaped counterpart to `align_corpus()`: reindexes `new`'s documents onto `old.vocab`, dropping out-of-vocabulary terms (and empty documents).

Usage

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

Arguments

`new`	An stm-style `list(documents, vocab)` or a `faSTM_corpus`.
`old.vocab`	Reference vocabulary to align onto.
`verbose`	Logical.

Value

`list(documents, vocab, docs.removed, words.removed)`.

ame — *Average marginal effects from an estimateEffect fit*

Description

The average expected change in a topic's proportion per unit of a covariate (continuous: average derivative; factor: average level-vs-reference contrast), averaged over the observed data. Cleaner than reading raw coefficients, especially with splines/interactions (cf. the `margins` package; stm #271).

Usage

```
ame(object, covariate, topics = object$topics, h = NULL, ci = 0.95)
```

Arguments

`object`	A `faSTM_effect` (from `estimateEffect()`).
`covariate`	Covariate name.
`topics`	Topics to report (default: all in the fit).
`h`	Step for the numeric derivative (continuous covariates); defaults to `0.01 * sd`.
`ci`	Confidence level.

Value

A data.frame: `topic`, `term`, `ame`, `se`, `lower`, `upper`.

`as.data.frame.faSTM_searchk` *Convert search_k diagnostics to long form for plotting*

Description

Returns a long data.frame (`K`, `metric`, `value`) ready for ggplot2 — `ggplot(as.data.frame(res), aes(K, value)) + geom_line() + facet_wrap(~metric, scales = "free_y")`.

Usage

```
## S3 method for class 'faSTM_searchk'
as.data.frame(x, ...)
```

Arguments

`x`	A `faSTM_searchk`.
`...`	Unused.

`as_corpus` *Build a faSTM corpus from prepared text*

Description

faSTM does not do its own tokenization — it reads an already-prepared document-term representation from the tools the field already uses (`quanteda`, `tidytext`) or a plain sparse matrix. `as_corpus()` normalizes any of these into the structure `stm()` consumes, dropping empty documents and re-indexing the vocabulary, with metadata kept aligned.

Usage

```
as_corpus(x, meta = NULL, ...)
```

Arguments

`x`	A `quanteda dfm`, a document-term `Matrix`/matrix (documents in rows, terms in columns, with `colnames`), or an existing `faSTM_corpus`. For a tidy (long) term table use `from_tidy()`.
`meta`	Optional data.frame of document metadata, one row per document, aligned to `x`. For a `dfm`, defaults to `quanteda::docvars(x)`.
`...`	Unused.

Value

A `faSTM_corpus`: a list with `documents` (named list of 2×n integer matrices: row 1 = 1-based term id, row 2 = count), `vocab` (character), `meta` (data.frame or NULL), and `word_counts` (corpus term frequencies).

---

asSTMCorpus *Coerce inputs into an stm-style corpus (stm-compatible)*

Description

Port of `stm::asSTMCorpus`'s role: accepts a `faSTM_corpus`, quanteda `dfm`, or document-term matrix and returns `list(documents, vocab, data)` in stm format.

Usage

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

Arguments

`documents`	A corpus/dfm/matrix, or an stm-style documents list.
`vocab`	Vocabulary (when `documents` is already a documents list).
`data`	Optional metadata.
`...`	Ignored.

Value

`list(documents, vocab, data)`.

---

augment.faSTM *Augment: most-likely topic for each document-term token*

Description

Assigns each (document, term) cell to the topic maximizing `theta[doc, k] * beta[k, term]` (cf. `tidytext::augment.STM`).

Usage

```
## S3 method for class 'faSTM'
augment(x, data = NULL, ...)
```

Arguments

`x`	A faSTM fit (carries its DTM).
`data`	Ignored (accepted for the generic).
`...`	Unused.

Value

A data.frame: `document`, `term`, `count`, `.topic`.

`calcfrex` *stm-compatible label scorers (FREX / lift / score)*

Description

Ports of `stm:::calcfrex/calclift/calcscore`. Each takes a K x V `logbeta` (log topic-word matrix) and returns a V x K matrix whose columns are the word indices ordered most- to least-characteristic for each topic.

Usage

```
calcfrex(logbeta, w = 0.5, wordcounts = NULL)

calclift(logbeta, wordcounts)

calcscore(logbeta)
```

Arguments

`logbeta`	K x V log topic-word matrix.
`w`	FREX frequency/exclusivity weight.
`wordcounts`	Corpus term frequencies (enables the James-Stein shrinkage).

Value

A V x K matrix of ordered word indices.

`check_residuals` *Residual dispersion check (is K large enough?)*

Description

Multinomial residual dispersion (Taddy 2012; port of `stm::checkResiduals`). A dispersion well above 1 suggests too few topics.

Usage

```
check_residuals(model, tol = 0.01)
```

Arguments

`model`	A faSTM fit (carries its documents).
`tol`	Threshold for counting estimable residual cells.

Value

A list with `dispersion`, `pvalue`, and `df`.

`checkBeta` *Flag words that load almost entirely on one topic*

Description

Port of `stm:::checkBeta`: finds (topic, word) cells whose `exp(logbeta)` exceeds `1 - tolerance` — words that are nearly exclusive to a single topic, which can destabilize estimation.

Usage

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

Arguments

`stmobject`	A faSTM/stm fit.
`tolerance`	Threshold; a word with topic-probability `> 1 - tolerance` is flagged.

Value

A list with `problemTopics`, `problemWords`, and error counts per content group.

`coherence` *Topic coherence (Mimno / NPMI / c_v)*

Description

Coherence scores for each topic's top-`M` words, computed from the fit's stored document-term matrix. `"mimno"` is the UMass-style score of `semantic_coherence()`; `"npmi"` averages pairwise normalized PMI; `"c_v"` is the Roeder et al. (2015) measure (one-set segmentation, NPMI confirmation, cosine aggregation). NPMI/c_v use *document* co-occurrence as the probability estimator. Higher is more coherent (npmi/c_v are roughly in -1, 1).

Usage

```
coherence(model, measure = c("mimno", "npmi", "c_v"), M = 10L)
```

Arguments

`model`	A faSTM fit (carries its DTM).
`measure`	`"mimno"`, `"npmi"`, or `"c_v"`.
`M`	Top words per topic.

Value

A numeric vector, one coherence score per topic.

`congress` *U.S. Congressional Speeches (Party x Chamber, 1987-2011)*

Description

A balanced sample of 1,679 floor speeches from the U.S. House and Senate, Congresses 100-111 (1987-2011). Speeches are sampled evenly across party x chamber x congress so covariate effects are estimable, then lowercased and pruned of stop words and rare terms. Metadata: `party` (Democrat/Republican), `chamber` (House/Senate), `congress` (100-111). Built to showcase multiple (crossed) content covariates and over-time prevalence.

Usage

```
congress
```

Format

A `faSTM_corpus` with 1,679 documents and a 4,110-term vocabulary.

Source

Congressional Record, Hein-bound edition (Gentzkow, Shapiro & Taddy), congresses 100-111. The underlying floor speeches are U.S. government works (public domain).

Examples

```
data(congress)
fit <- stm(congress, K = 12, prevalence = ~ party + s(congress),
           content = ~ party + chamber)
```

`content_topics` *Marginal content words by one content covariate*

Description

For a multi-covariate (crossed) content model, recovers the topic-word labels for each level of a single content covariate, averaging the crossed topic-word distributions over the other covariate(s). Lets you read off how topics' vocabulary shifts with one covariate while marginalizing the rest.

Usage

```
content_topics(model, by = NULL, n = 7L, type = c("prob", "lift", "frex"))
```

Arguments

`model`	A content (SAGE) faSTM fit.
`by`	Content covariate name to marginalize *to* (default: the first).
`n`	Words per topic.
`type`	`"prob"`, `"lift"`, or `"frex"`.

Value

A named list (one entry per level of `by`) of K x `n` word matrices.

`convertCorpus` *Convert documents/vocab between corpus formats (stm-compatible)*

Description

Port of `stm:::convertCorpus`. `"Matrix"` returns a documents x V sparse dgCMatrix; `"lda"` returns the documents list (the lda/stm format).

Usage

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

Arguments

`documents`	stm-style documents list.
`vocab`	Vocabulary vector.
`type`	`"Matrix"` or `"lda"`.

Value

The corpus in the requested format.

`effect_estimates` *Extract estimateEffect estimates as a tidy data.frame (no plotting)*

Description

Returns the point estimates, standard errors and confidence bounds that `plot.faSTM_effect()` would draw, so you can build a custom plot or table (stm issue #83). Same arguments as the plot method.

Usage

```
effect_estimates(
  x,
  covariate,
  method = c("pointestimate", "continuous", "difference"),
  topics = x$topics,
  cov.value1 = NULL,
  cov.value2 = NULL,
  values = NULL,
  moderator = NULL,
  moderator.value = NULL,
  npoints = 50L,
  ci = 0.95
)
```

Arguments

`x`	A `faSTM_effect` (from `estimateEffect()`).
`covariate`	Covariate name.
`method`	`"pointestimate"`, `"continuous"`, or `"difference"`.
`topics`	Topics to include.
`cov.value1`, `cov.value2`, `values`	Levels/values for difference or continuous range.
`moderator`, `moderator.value`	Optional held-fixed interaction term.
`npoints`	Grid size for `"continuous"`.
`ci`	Confidence level for `lower`/`upper`.

Value

A data.frame with `topic`, `value`, `est`, `se`, `lower`, `upper`.

`estimateEffect` *Estimate covariate effects on topic prevalence (method of composition)*

Description

A drop-in for `stm::estimateEffect()` that propagates per-document topic-estimation uncertainty: it regresses each posterior draw of topic proportions on the covariates and pools the per-draw fits by Rubin's rules. Propagating that uncertainty is the reason faSTM ships its own estimator rather than inheriting stm's.

Usage

```
estimateEffect(
  formula,
  stmobj,
  metadata = meta,
  uncertainty = c("Global", "None", "Local"),
  nsims = 100L,
  seed = NULL,
  meta = NULL,
  documents = NULL,
  combine = NULL,
  weights = NULL,
  cluster = NULL,
  ...
)
```

Arguments

`formula`	A formula whose LHS lists topic numbers (e.g. `1:5 ~ treatment`) or whose LHS is empty to use all topics; RHS gives the covariates. Random-effect terms `(term | group)` are supported (fits `lme4::lmer` per draw and pools the fixed effects; variance components are stored).
`stmobj`	A faSTM fit (from `stm()`).
`metadata`	A data.frame of covariates aligned to the documents.
`uncertainty`	`"Global"` (method of composition over posterior draws, default) or `"None"` (single OLS on the posterior-mean theta).
`nsims`	Posterior draws for `uncertainty = "Global"`.
`seed`	Optional seed for the posterior draws.
`documents`	Accepted for stm compatibility (faSTM reads nu from the fit).
`combine`	Optional list of topic vectors to also estimate as aggregate topics (each set's proportions are summed before regressing); named entries set the coefficient names. E.g. `combine = list(econ = c(3, 7))`.
`weights`	Optional per-document survey/sampling weights (weighted OLS).

`cluster`	Optional per-document cluster ids for cluster-robust SEs.
`...`	Unused (stm signature compatibility).

Value

An object of class `c("faSTM_effect", "estimateEffect")` with a `summary()` method, holding pooled coefficients and standard errors per topic.

`eval_heldout` *Evaluate held-out log-likelihood of a fit on a held-out set*

Description

Evaluate held-out log-likelihood of a fit on a held-out set

Usage

```
eval_heldout(model, heldout)
```

Arguments

`model`	A faSTM fit (trained on `heldout$corpus`).
`heldout`	A `faSTM_heldout` (or its `missing` list).

Value

Mean per-document held-out log-likelihood per token.

`exclusivity` *Topic exclusivity (FREX-summary, frexw default 0.7)*

Description

Topic exclusivity (FREX-summary, frexw default 0.7)

Usage

```
exclusivity(model, M = 10L, frexw = 0.7)
```

Arguments

`model`	A faSTM fit.
`M`	Top words per topic.
`frexw`	Frequency/exclusivity weight.

Value

A numeric vector, one exclusivity value per topic.

find_thoughts *Representative documents for each topic*

Description

Representative documents for each topic

Usage

```
find_thoughts(model, texts = NULL, topics = NULL, n = 3L)
```

Arguments

model	A faSTM fit.
texts	Optional character vector of the raw document texts, aligned to the fitted documents; returned alongside the indices when supplied.
topics	Topics to report (default all).
n	Documents per topic.

Value

A list with `index` (per-topic document indices) and, if `texts` is given, `docs` (the texts).

find_topic *Find topics whose top words include given words*

Description

Find topics whose top words include given words

Usage

```
find_topic(model, words, n = 20L, type = c("prob", "frex", "lift", "score"))
```

Arguments

model	A faSTM fit.
words	Character vector of query words.
n	Top words per topic to search.
type	Ranking metric: `"prob"`, `"frex"`, `"lift"`, or `"score"`.

Value

Integer vector of matching topics.

`fit_new_documents` *Infer topic proportions for new documents*

Description

Runs the variational E-step for each new document against the fitted model's fixed global parameters (topic-word matrix, prior mean and covariance), giving out-of-sample topic proportions (cf. `stm::fitNewDocuments`). The model's topics are held fixed; only each new document's proportions are estimated.

Usage

```
fit_new_documents(model, newdata)
```

Arguments

`model`	A faSTM fit (non-content; for content models the group-marginal topic-word matrix is used, with a warning).
`newdata`	A `faSTM_corpus`, quanteda `dfm`, or document-term matrix. Terms are aligned to the model's vocabulary; out-of-vocabulary terms are dropped.

Value

A new-documents × K matrix of topic proportions.

`fit_stm` *Fit a structural topic model and return its raw arrays.*

Description

Inputs are pre-converted in `R/stm.R`:

- `docs_flat` / `doc_lens`: documents as one concatenated 0-based token-id stream plus per-document lengths (counts already expanded). Reassembled here into `Vec<Vec<u32>>`, the shape `fit_ctm` wants.
- `prevalence`: row-major D×P design matrix flattened (NULL if none).
- `content_groups`: per-doc 0-based group id (NULL if none); `num_groups`.

Usage

```
fit_stm(
  docs_flat,
  doc_lens,
  num_types,
  num_topics,
  em_iters,
  em_tol,
  sigma_shrink,
  prevalence,
  num_features,
  content_groups,
  num_groups,
  init_spectral,
  init_beta,
  gamma_l1_alpha,
  diagonal,
  seed,
  inference,
  batch_size,
  tau,
  kappa,
  num_threads
)
```

Details

`inference`: "batch" -> `fit_ctm` (parity-validated). "svi" -> `fit_ctm_svi` once topica #231 PR B (STM-SVI) is in the pinned revision; the R layer gates the prevalence/content + svi combination until then.

`fitNewDocuments` *Infer topics for new documents (stm-compatible signature)*

Description

Drop-in for `stm::fitNewDocuments()`. Holds the fitted topics fixed and runs the variational E-step for each new document. Supports stm's prior modes and posterior return.

Usage

```
fitNewDocuments(
  model,
  documents,
  newData = NULL,
  origData = NULL,
  prevalence = NULL,
```

```
  betaIndex = NULL,
  prevalencePrior = c("Average", "Covariate", "None"),
  contentPrior = c("Covariate", "Average"),
  returnPosterior = FALSE,
  verbose = TRUE,
  ...
)
```

Arguments

`model`	A faSTM fit.
`documents`	New documents: a `faSTM_corpus`/`dfm`/matrix (aligned to the model vocabulary), or an stm-style list of 2 x n integer matrices indexed into `model$vocab`.
`newData`, `origData`	Covariate frames for the new and original documents (used by `prevalencePrior = "Covariate"` to set each document's prior mean).
`prevalence`	Prevalence formula (same RHS as the fit) for the covariate prior.
`betaIndex`	Integer per-document content-group index (content models).
`prevalencePrior`	`"Average"` (global prior mean, default) or `"Covariate"` (per-document mean from `prevalence`/`newData`).
`contentPrior`	`"Covariate"` (use the group's topic-word matrix via `betaIndex`, default) or `"Average"` (group-marginal).
`returnPosterior`	If `TRUE`, return `list(theta, eta, nu)` (per-document variational mean and Laplace covariance); otherwise a documents x K theta matrix.
`verbose`	Logical.
`...`	Ignored (stm signature compatibility).

Value

A theta matrix, or a posterior list when `returnPosterior = TRUE`.

`frex_scores` *FREX scores for every word and topic*

Description

FREX balances word *frequency* and *exclusivity* (Bischof & Airoldi 2012; Roberts et al.). Unlike `stm`'s `labelTopics()`, this returns the full numeric FREX matrix, not just the ranked words (addresses a long-standing `stm` request, bstewart/stm#265).

Usage

```
frex_scores(model, w = 0.5)
```

Arguments

`model`	A faSTM fit.
`w`	FREX frequency/exclusivity weight (0.5 = equal).

Value

A topics × vocabulary matrix of FREX scores (columns named by vocab).

`from_tidy` *Build a faSTM corpus from a tidy (long) term-count table*

Description

For `tidytext`-style data: one row per (document, term) with a count.

Usage

```
from_tidy(data, document = "document", term = "term", count = "n", meta = NULL)
```

Arguments

`data`	A data.frame.
`document`, `term`, `count`	Column names (strings) for the document id, the term, and the count. `count` defaults to a count of rows per (doc, term).
`meta`	Optional per-document metadata, aligned to the sorted unique documents.

Value

A `faSTM_corpus`.

`glance.faSTM` *One-row model summary for a faSTM fit*

Description

One-row model summary for a faSTM fit

Usage

```
## S3 method for class 'faSTM'
glance(x, ...)
```

Arguments

x	A faSTM fit.
...	Unused.

Value

A one-row data.frame.

`infer_theta_new` *Out-of-sample topic inference: for each new document, run the variational E-step against fixed globals (, , Σ ⁻¹) and return . Documents are passed sparse —* `words` *are 0-based ids into the* fitted model's *vocabulary (out-of-vocabulary terms dropped by the R caller) with their* `counts`*, concatenated, plus per-document term counts* `doc_nterms`*.*

Description

Out-of-sample topic inference: for each new document, run the variational E-step against fixed globals (, , Σ ⁻¹) and return . Documents are passed sparse — `words` are 0-based ids into the *fitted model's* vocabulary (out-of-vocabulary terms dropped by the R caller) with their `counts`, concatenated, plus per-document term counts `doc_nterms`.

Usage

```
infer_theta_new(
  beta_flat,
  num_topics,
  num_types,
  mu,
  siginv,
  words,
  counts,
  doc_nterms
)
```

`label_topics` *Label topics by top words (prob, FREX, lift, score)*

Description

Label topics by top words (prob, FREX, lift, score)

Usage

```
label_topics(model, n = 7L, frexweight = 0.5)
```

Arguments

`model`	A faSTM fit.
`n`	Number of words per topic per metric.
`frexweight`	FREX frequency/exclusivity weight.

Value

A `faSTM_labels` object: per-metric top-word matrices (`prob`, `frex`, `lift`, `score`), each topics × `n`.

`lda_init_beta` *LDA topic-word matrix via topica's CVB0 (deterministic collapsed variational Bayes), to seed a "replicate stm's LDA init" STM fit. Mirrors stm's collapsed-Gibbs LDA initialization; the result is fed back as* `init_beta`*. Returns K*V row-major topic-word probabilities.*

Description

LDA topic-word matrix via topica's CVB0 (deterministic collapsed variational Bayes), to seed a "replicate stm's LDA init" STM fit. Mirrors stm's collapsed-Gibbs LDA initialization; the result is fed back as `init_beta`. Returns K*V row-major topic-word probabilities.

Usage

```
lda_init_beta(
  docs_flat,
  doc_lens,
  num_types,
  num_topics,
  iters,
  alpha,
  beta,
  seed
)
```

`make_dt` *Document-topic proportions as a data frame*

Description

Document-topic proportions as a data frame

Usage

```
make_dt(model, meta = NULL)
```

Arguments

`model`	A faSTM fit.
`meta`	Optional metadata to bind alongside (defaults to none).

Value

A data.frame with `document` and `Topic1..TopicK` columns (+ `meta`).

`make_heldout` *Create a held-out version of a corpus for document-completion validation*

Description

Create a held-out version of a corpus for document-completion validation

Usage

```
make_heldout(
  corpus,
  N = floor(0.1 * length(corpus$documents)),
  proportion = 0.5,
  seed = NULL
)
```

Arguments

`corpus`	A `faSTM_corpus`.
`N`	Number of documents to hold tokens out of (default: 10% of docs).
`proportion`	Fraction of each chosen document's term *types* to hold out.
`seed`	Optional RNG seed.

Value

A list with `corpus` (training corpus, held-out tokens removed) and `missing` (per-document held-out terms + counts), class `faSTM_heldout`.

makeDesignMatrix *Build a (sparse) design matrix for new data (stm-compatible)*

Description

Port of `stm:::makeDesignMatrix`: builds the model matrix for `newData` using the term structure and factor levels of `origData`.

Usage

```
makeDesignMatrix(formula, origData, newData, sparse = TRUE, ...)
```

Arguments

`formula`	A model formula.
`origData`	Data defining the terms/levels.
`newData`	Data to build the matrix for.
`sparse`	Return a sparse matrix.
`...`	Ignored.

Value

A (sparse) design matrix.

many_topics *Select models across a range of K*

Description

Runs `select_model()` for each K and returns the chosen model per K.

Usage

```
many_topics(
  corpus,
  K,
  N = 10L,
  prevalence = NULL,
  content = NULL,
  by = "sum",
  cores = 1L,
  seed = 1L,
  ...
)
```

Arguments

`corpus`	A `faSTM_corpus`.
`K`	Integer vector of topic counts.
`N`	Number of candidate models (distinct random inits).
`prevalence`, `content`	Optional covariate formulas.
`by`	Selection rule passed to `select_best()`.
`cores`	Candidates to fit in parallel.
`seed`	Base RNG seed (candidate i uses `seed + i - 1`).
`...`	Passed to `stm()`.

Value

A `faSTM_manytopics`: `models` (best per K) and a `summary` data.frame.

`multi_stm` *Cross-run topic stability*

Description

Aligns every model from a `select_model()` run to the first and reports how stable each topic's top words are across runs (cf. `stm::multiSTM`).

Usage

```
multi_stm(x, n = 10L)
```

Arguments

`x`	A `faSTM_selectmodel`.
`n`	Top words used for the stability score.

Value

A `faSTM_multistm` with a per-topic mean top-word agreement.

optimizeDocument *Per-document variational E-step (stm-compatible)*

Description

Port of `stm:::optimizeDocument`'s interface: infers one document's topic proportions against fixed globals and returns its variational mean `lambda` (eta), Laplace covariance `nu`, and `theta`.

Usage

```
optimizeDocument(document, eta, mu, beta, sigma = NULL, sigmainv = NULL, ...)
```

Arguments

`document`	A 2 x n integer matrix (1-based vocab ids; counts).
`eta`	Ignored starting value (kept for signature compatibility).
`mu`	Prior mean (length K-1).
`beta`	K x V topic-word probability matrix.
`sigma`, `sigmainv`	Prior covariance or its inverse (supply one).
`...`	Ignored (stm signature compatibility).

Value

A list with `lambda`, `nu`, and `theta`.

permutation_test *Permutation test for a binary covariate's effect on topics*

Description

Refits the model many times with the treatment labels permuted, aligning topics across refits, to build a null distribution for the treatment effect on each topic (cf. `stm::permutationTest`). Fast because each refit is cheap.

Usage

```
permutation_test(
  formula,
  model,
  treatment,
  corpus,
  nruns = 100L,
  seed = NULL,
  ...
)
```

Arguments

`formula`	Prevalence formula whose RHS includes `treatment`.
`model`	A faSTM fit.
`treatment`	Name of a 0/1 covariate in `corpus$meta`.
`corpus`	The `faSTM_corpus` the model was fit on.
`nruns`	Total models (1 reference + `nruns-1` permutations).
`seed`	RNG seed.
`...`	Passed to `stm()` for the refits.

Value

A `faSTM_permtest` with `ref` (observed per-topic effects) and `null` (`(nruns-1)` × K permuted effects).

`plot.faSTM` *Plot a fitted model*

Description

Plot a fitted model

Usage

```
## S3 method for class 'faSTM'
plot(
  x,
  type = c("summary", "labels", "perspectives", "hist"),
  topics = NULL,
  n = 5L,
  labeltype = "frex",
  ...
)
```

Arguments

`x`	A faSTM fit.
`type`	`"summary"` (topics ranked by expected prevalence + top words), `"labels"` (top words per topic), `"perspectives"` (word comparison between two topics, or between content-covariate groups of one topic), or `"hist"` (distribution of document-topic proportions).
`topics`	Topics to show (for `"perspectives"`: one topic in a content model, or two topics to compare).
`n`	Top words to label each topic with.
`labeltype`	Word ranking for labels: `"frex"`, `"prob"`, `"lift"`, `"score"`.
`...`	Accepted for stm compatibility (e.g. `xlim`); mostly ignored.

Value

A ggplot object.

`plot.faSTM_effect` *Plot estimated covariate effects on topic prevalence*

Description

Plot estimated covariate effects on topic prevalence

Usage

```
## S3 method for class 'faSTM_effect'
plot(
  x,
  covariate,
  method = c("pointestimate", "continuous", "difference"),
  topics = x$topics,
  model = NULL,
  cov.value1 = NULL,
  cov.value2 = NULL,
  values = NULL,
  moderator = NULL,
  moderator.value = NULL,
  npoints = 50L,
  ci = 0.95,
  labeltype = NULL,
  custom.labels = NULL,
  xlab = NULL,
  main = NULL,
  ...
)
```

Arguments

`x`	A `faSTM_effect` (from `estimateEffect()`).
`covariate`	Name of the covariate to vary.
`method`	`"pointestimate"` (mean proportion per level of a categorical covariate), `"continuous"` (proportion vs a numeric covariate, with ribbon), or `"difference"` (difference between two `values`).
`topics`	Topics to show (default all in the effect object).
`values`	For `"difference"`, length-2 `c(high, low)`; for `"continuous"`, optional range; ignored for `"pointestimate"`.
`npoints`	Grid size for `"continuous"`.
`ci`	Confidence level.
`...`	Unused.

Value

A ggplot object.

`plot.faSTM_searchk` *Plot search_k diagnostics*

Description

Faceted held-out likelihood, semantic coherence, exclusivity and bound vs K.

Usage

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

Arguments

`x`	A `faSTM_searchk`.
`...`	Unused.

Value

A ggplot object.

`plot_topic_network` *Topic correlation network*

Description

Nodes are topics (sized by prevalence, labelled by top words); edges join topics whose proportions are positively correlated above `cutoff`. Uses a lightweight circular layout — no graph-library dependency.

Usage

```
plot_topic_network(model, cutoff = 0.03, n = 3L, labeltype = "frex")
```

Arguments

`model`	A faSTM fit.
`cutoff`	Correlation threshold for an edge.
`n`	Top words per topic label.
`labeltype`	Word ranking for labels.

Value

A ggplot object.

poliblog *CMU 2008 Political Blog Corpus (poliblog5k)*

Description

5,000 political blog posts from the 2008 U.S. election (the **stm** vignette example) as a ready-to-use `faSTM_corpus`. Metadata: `rating` (Conservative/Liberal), `day` (1-365), `blog`, `text`.

Usage

```
data(poliblog)
```

Format

A `faSTM_corpus`: 5,000 documents, 2,632-term vocabulary.

Source

Eisenstein & Xing (2010), via the **stm** package.

posterior_theta_samples *Draw from the per-document topic-proportion posterior*

Description

The variational (Laplace) posterior of each document's logit-topic vector is `eta_d ~ N(lambda_d, nu_d)`, both stored on a faSTM fit. This draws `nsims` samples of theta per document by sampling eta and applying the softmax (with the reference topic appended as 0). This is the pure-R equivalent of topica's `posterior_theta_samples`; no Rust call is needed because `eta` + `nu` fully describe the posterior. Feeds `estimateEffect()`'s method of composition.

Usage

```
posterior_theta_samples(model, nsims = 100L, seed = NULL)
```

Arguments

`model`	A faSTM fit (from `stm()`).
`nsims`	Number of posterior draws.
`seed`	Optional integer seed for reproducible draws.

Value

A `nsims`-length list of `D x K` theta matrices.

`predict.faSTM` *Predict topic proportions for new documents*

Description

Predict topic proportions for new documents

Usage

```
## S3 method for class 'faSTM'
predict(object, newdata, ...)
```

Arguments

`object`	A faSTM fit.
`newdata`	New documents (corpus / dfm / matrix / stm-style list).
`...`	Passed to `fit_new_documents()`.

Value

A new-documents x K matrix of topic proportions.

`read_ldac` *Read/write a corpus in LDA-C (Blei) sparse format*

Description

Each line is `M term:count term:count ...` with 0-based term ids.

Usage

```
read_ldac(file)

write_ldac(documents, file)
```

Arguments

`file`	Path to the `.ldac`/`.dat` file (read) or output path (write).
`documents`	A list of 2×n integer matrices (1-based ids).

Value

`read_ldac` returns a list of 2×n integer matrices (faSTM/stm document format, 1-based ids); `write_ldac` returns the path invisibly.

s — *Spline term for prevalence formulas*

Description

A b-spline basis for smooth covariate effects, e.g. `prevalence = ~ s(day)`. Matches `stm::s()` exactly — including the `df = min(10, nval - 1)` default — so spline-term coefficients agree with `stm`. (You can also use `splines::bs()`/`splines::ns()` directly.)

Usage

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

Arguments

`x`	Numeric predictor.
`df`	Basis dimension; defaults to `min(10, length(unique(x)) - 1)`.
`...`	Passed to `splines::bs()`.

Value

A spline basis matrix (with class `"s"`).

sage_labels — *Labels for a content (SAGE) model*

Description

For models fit with a `content` covariate, reports each topic's marginal top words plus, for every content group, the words most distinctive to that group within the topic (group-vs-marginal log-ratio — the SAGE deviation).

Usage

```
sage_labels(model, n = 7L, frexweight = NULL)
```

Arguments

`model`	A faSTM fit with a content covariate.
`n`	Words per list.

Value

A `faSTM_sagelabels` object.

`search_k` *Search over the number of topics K*

Description

Fits the model across a range of K and reports diagnostics for choosing it: held-out likelihood (document completion), semantic coherence, exclusivity, and the variational bound. Unlike `stm::searchK`, the per-K fits parallelize across K (a long-standing request, bstewart/stm#262) and each fit is itself fast (Rust), so a sweep that took minutes takes seconds.

Usage

```
search_k(
  corpus,
  K,
  prevalence = NULL,
  content = NULL,
  heldout = TRUE,
  proportion = 0.5,
  residuals = FALSE,
  cores = 1L,
  M = 10L,
  seed = 1L,
  measure = c("mimno", "npmi", "c_v"),
  verbose = FALSE,
  ...
)
```

Arguments

`corpus`	A `faSTM_corpus` (from `as_corpus()`).
`K`	Integer vector of topic counts to try.
`prevalence`, `content`	Optional covariate formulas (see `stm()`).
`heldout`	Logical; compute held-out likelihood via document completion.
`proportion`	Held-out token fraction (passed to `make_heldout()`).
`cores`	Number of K-fits to run in parallel (forked; 1 = sequential). When `cores > 1` each fit runs single-threaded to avoid oversubscription; when `cores == 1` each fit uses all cores.
`M`	Top words for coherence/exclusivity.
`seed`	RNG seed (held-out split + fits).
`...`	Passed to `stm()` (e.g. `max.em.its`, `init.type`).

Value

A `faSTM_searchk` object wrapping a tidy data.frame `results` with one row per K (`K`, `heldout`, `semcoh`, `exclusivity`, `bound`).

---

select_best *Pick one model from a* `select_model` *run*

Description

Pick one model from a `select_model` run

Usage

```
select_best(x, by = c("sum", "semcoh", "exclusivity"))
```

Arguments

`x`	A `faSTM_selectmodel`.
`by`	`"semcoh"`, `"exclusivity"`, or `"sum"` (rank-sum of both).

Value

A single faSTM fit.

---

select_model *Fit several models and keep the ones on the quality frontier*

Description

With random initialization the variational objective is multimodal, so the standard workflow (cf. `stm::selectModel`) is to fit many models and keep those on the semantic-coherence / exclusivity frontier, then choose among them. faSTM fits the candidates in parallel.

Usage

```
select_model(
  corpus,
  K,
  N = 10L,
  prevalence = NULL,
  content = NULL,
  init.type = "Random",
  cores = 1L,
  M = 10L,
  frexw = 0.7,
  seed = 1L,
  ...
)
```

Arguments

`corpus`	A `faSTM_corpus`.
`K`	Number of topics.
`N`	Number of candidate models (distinct random inits).
`prevalence`, `content`	Optional covariate formulas.
`init.type`	Initialization; `"Random"` (the point of selecting) or `"Spectral"` (deterministic — then all `N` are identical).
`cores`	Candidates to fit in parallel.
`M`	Top words for coherence/exclusivity scoring.
`frexw`	Exclusivity FREX weight.
`seed`	Base RNG seed (candidate i uses `seed + i - 1`).
`...`	Passed to `stm()`.

Value

A `faSTM_selectmodel`: `models` (the fits), `semcoh`, `exclusivity`, and `frontier` (indices of non-dominated models).

`semantic_coherence` *Semantic coherence (Mimno et al. 2011)*

Description

Sum over the top-`M` words of each topic of `log((D(w_i,w_j)+1)/D(w_j))`, using document co-occurrence counts. Higher (less negative) is more coherent.

Usage

```
semantic_coherence(model, M = 10L)
```

Arguments

`model`	A faSTM fit (must carry its document-term matrix; faSTM stores it).
`M`	Number of top words per topic.

Value

A numeric vector, one coherence value per topic.

`stm` *Fit a structural topic model (fast Rust backend, stm-compatible object)*

Description

A drop-in replacement for `stm::stm()`'s fitting step. Accepts the same `documents` / `vocab` / `prevalence` / `content` inputs, fits with topica's Rust core, and returns an object compatible with the `stm` package so that `stm::labelTopics()`, `stm::plot.STM()`, `stm::findThoughts()`, `stm::sageLabels()`, and `stm::toLDAvis()` work unmodified. Use `estimateEffect()` from this package for covariate effects that propagate topic-estimation uncertainty.

Usage

```
stm(
  documents,
  vocab,
  K,
  prevalence = NULL,
  content = NULL,
  data = NULL,
  max.em.its = 500L,
  emtol = 1e-05,
  init.type = c("Spectral", "Random", "LDA", "Custom"),
  init.beta = NULL,
  model = NULL,
  gamma.prior = c("Pooled", "L1"),
  gamma.l1.alpha = 0.001,
  sigma.prior = 0,
  seed = 1L,
  inference = c("batch", "svi"),
  batch_size = 256L,
  tau = 64,
  kappa = 0.7,
  num_threads = 0L,
  verbose = TRUE,
  ...
)
```

Arguments

`documents`	stm-format documents: a named list of `2 x n_d` integer matrices (row 1 = 1-based word id into `vocab`, row 2 = count). Produced by `stm::prepDocuments()`.
`vocab`	Character vector of vocabulary terms.
`K`	Number of topics.

`prevalence`	A right-hand-side formula (e.g. `~ treatment + s(age)`) or a design matrix; topic prevalence covariates. `data` supplies the variables.
`content`	A right-hand-side formula naming a single categorical variable, or a factor; the SAGE content covariate. `data` supplies the variable.
`data`	A data.frame of document metadata (the `meta` from `stm::prepDocuments()`), aligned to `documents`.
`max.em.its`	Maximum EM iterations (batch) / epochs (svi).
`emtol`	Relative-bound convergence tolerance.
`init.type`	Topic initialization: `"Spectral"` (stm's default), `"Random"`, `"LDA"` (seed from a quick CVB0 LDA, like stm's collapsed-Gibbs init), or `"Custom"` (seed from `init.beta` or a supplied `model`).
`init.beta`	Optional K x V topic-word probability matrix to start the fit from a given initialization (overrides `init.type`). Supplying R `stm`'s exact spectral beta here reproduces that run — a guaranteed "replicate the original" mode (topica #234/#235).
`model`	A fitted model whose topic-word matrix seeds `init.type = "Custom"`.
`gamma.prior`	Prevalence-coefficient prior: `"Pooled"` (ridge, stm default) or `"L1"`.
`sigma.prior`	Shrinkage applied to the topic covariance off-diagonal.
`seed`	Integer seed (batch fit is reproducible from it).
`inference`	`"batch"` (default, parity-validated) or `"svi"` (stochastic variational; scales to large corpora — requires a topica build with STM-SVI).
`batch_size`, `tau`, `kappa`	SVI controls (minibatch size; Robbins-Monro `(tau + t)^(-kappa)` step schedule). Ignored when `inference = "batch"`.
`num_threads`	Worker threads for the parallel variational E-step. `0` (default) uses all cores; `>= 1` pins a scoped pool. Results are identical regardless of thread count.
`verbose`	Logical; print progress.

Value

An object of class `c("faSTM", "STM")` — an stm-compatible fit.

`tidy.faSTM` *Tidy a faSTM fit (topic-term or document-topic distributions)*

Description

Tidy a faSTM fit (topic-term or document-topic distributions)

Usage

```
## S3 method for class 'faSTM'
tidy(x, matrix = c("beta", "gamma", "frex"), ...)
```

Arguments

`x`	A faSTM fit.
`matrix`	`"beta"` (topic-term probabilities), `"gamma"` (document-topic proportions), or `"frex"` (topic-term FREX scores).
`...`	Unused.

Value

A tidy data.frame.

`tidy.faSTM_effect` *Tidy an estimateEffect fit (one row per term per topic)*

Description

Tidy an estimateEffect fit (one row per term per topic)

Usage

```
## S3 method for class 'faSTM_effect'
tidy(x, ...)
```

Arguments

`x`	A `faSTM_effect`.
`...`	Unused.

Value

A data.frame: `topic`, `term`, `estimate`, `std.error`, `statistic`, `p.value`.

`topic_corr_graph` *Topic-correlation network as an igraph graph*

Description

Exports the positive-correlation topic network as an `igraph` object (stm issue #242), with topic prevalence and FREX labels as vertex attributes and the positive correlations as edge weights — ready for igraph/ggraph layouts.

Usage

```
topic_corr_graph(x, model = NULL, nlabel = 3L)
```

Arguments

x	A faSTM_topiccorr (from topicCorr()) or a faSTM fit.
model	The fit, if x is a bare correlation object (for vertex prevalence/labels).
nlabel	Top FREX words per topic for the vertex label.

Value

An undirected igraph graph.

---

topic_correlation *Topic correlation graph (positive correlations of topic proportions)*

Description

Topic correlation graph (positive correlations of topic proportions)

Usage

```
topic_correlation(model, cutoff = 0.01)
```

Arguments

model	A faSTM fit.
cutoff	Correlation threshold for an edge.

Value

A list with cor (the K×K correlation matrix) and posadj (the thresholded positive adjacency).

---

topic_lasso *Predict a document-level outcome from topic proportions (lasso)*

Description

Cross-validated lasso (glmnet) of an outcome on the topic-proportion matrix (cf. stm::topicLasso). Identifies which topics predict the outcome.

Usage

```
topic_lasso(
  formula,
  model,
  data,
  family = "gaussian",
  nfolds = 10L,
  seed = 2138L,
  ...
)
```

Arguments

`formula`	`outcome ~ .` — the LHS names the outcome in `data`.
`model`	A faSTM fit (supplies the topic proportions).
`data`	Document-level data with the outcome, aligned to the documents.
`family`	glmnet family (`"gaussian"`, `"binomial"`, ...).
`nfolds`	CV folds.
`seed`	RNG seed.
`...`	Passed to `glmnet::cv.glmnet()`.

Value

A `faSTM_topiclasso` with selected per-topic coefficients.

`topic_proportions` *Expected topic proportions (the numbers behind the summary plot)*

Description

Returns the corpus-level expected topic proportions — the mean of theta per topic — as a numeric table, so you can read off the values stm's `plot(type = "summary")` displays (stm issue #269).

Usage

```
topic_proportions(model, nlabel = 3L)
```

Arguments

`model`	A faSTM fit.
`nlabel`	Top FREX words to attach as a topic label.

Value

A data.frame with `topic`, `proportion`, `label`, sorted by proportion.

`topic_terms` *Top terms per topic, with their numeric scores (tidy)*

Description

Like `label_topics()` but returns the *values* behind the ranking, not just the words — e.g. the numeric FREX score per top term (stm issue #265).

Usage

```
topic_terms(
  model,
  n = 7L,
  by = c("prob", "frex", "lift", "score"),
  frexweight = 0.5
)
```

Arguments

`model`	A faSTM fit.
`n`	Terms per topic.
`by`	Ranking measure: `"prob"`, `"frex"`, `"lift"`, or `"score"`.
`frexweight`	FREX frequency/exclusivity weight (used when `by = "frex"`).

Value

A tidy data.frame with `topic`, `rank`, `term`, `score`, `measure`.

Index