Package 'tmfast'

May 30, 2026

Title Fast Topic Models Using Varimax

Version 0.1.1

Description Fits topic models using varimax-rotated principal component analysis (PCA), following the ``vintage factor analysis'' approach of Rohe & Zheng (2020) <doi:10.48550/arXiv.2004.05387>. Leverages truncated PCA via 'irlba' for sparse matrices, enabling fast model fitting on large corpora. Includes an information-theoretic approach to vocabulary selection, 'broom'-compatible tidiers for extracting word-topic and topic-document matrices into a tidy data workflow, and samplers for constructing simulated corpora for benchmarking and method evaluation.

License GPL (>= 3)

Encoding UTF-8

RoxygenNote 7.3.2

Imports assertthat, purrr, dplyr, tidyr, magrittr, rlang, stringr, tibble, tidyselect, irlba, tidytext, glue, Matrix, generics, psych, cli

Suggests testthat (>= 3.0.0), knitr, rmarkdown, ggbeeswarm, ggplot2, Rtsne, umap, lpSolve, janeaustenr, stm, tictoc, furrr, reshape2, tmfast.realbooks

Additional_repositories https://dhicks.github.io/drat/

VignetteBuilder knitr

Config/testthat/edition 3

URL https://dhicks.github.io/tmfast/, https://github.com/dhicks/tmfast

BugReports https://github.com/dhicks/tmfast/issues

NeedsCompilation no

Author D. Hicks [aut, cre, cph] (ORCID: <https://orcid.org/0000-0001-7945-4416>)

Maintainer D. Hicks <hicks.daniel.j@gmail.com>

Depends R (>= 4.1.0)

Repository CRAN

Date/Publication 2026-05-30 13:40:02 UTC

Contents

`tmfast-package` *Fitting "topic models" with PCA+varimax*

Description

Fits topic models using varimax-rotated principal component analysis (PCA), following the "vintage factor analysis" approach of Rohe & Zheng (2020) doi:10.48550/arXiv.2004.05387. Leverages truncated PCA via 'irlba' for sparse matrices, enabling fast model fitting on large corpora. Includes an information-theoretic approach to vocabulary selection, 'broom'-compatible tidiers for extracting word-topic and topic-document matrices into a tidy data workflow, and samplers for constructing simulated corpora for benchmarking and method evaluation.

Author(s)

Maintainer: D. Hicks <hicks.daniel.j@gmail.com> (ORCID) [copyright holder]

See Also

Useful links:

- https://dhicks.github.io/tmfast/
- https://github.com/dhicks/tmfast
- Report bugs at https://github.com/dhicks/tmfast/issues

`build_matrix` *Convert a long dataframe to a wide (sparse) matrix*

Description

For the sparse case, an alias for `tidytext::cast_sparse`

Usage

```
build_matrix(data, row, column, value, ..., sparse = TRUE)
```

Arguments

`data`	Dataframe
`row`	Column name to use as row names, as string or symbol
`column`	Column name to use as column names, as string or symbol
`value`	Column name to use as matrix values, as string or symbol
`...`	Other arguments, passed to `Matrix::sparseMatrix`
`sparse`	Should the matrix be a `Matrix` sparse matrix?

Value

A matrix or sparse Matrix object, with one row for each unique value in the row column, one column for each unique value in the column column, and with as many non-zero values as there are rows in data.

Examples

```
data.frame(id = c(1, 1, 2, 2) + 4,
           cols = c('a', 'b', 'a', 'b'),
           vals = 1:4) |>
    build_matrix(row = id, column = 'cols', value = vals)
```

`compare_betas` *Compare topic-word distributions using Hellinger distance*

Description

Computes pairwise Hellinger distances between topics from one or two fitted models. Tokens missing from a beta dataframe are filled with probability 0 before comparison, so both models need not share the same vocabulary.

Usage

```
compare_betas(beta1, beta2 = NULL, vocab)
```

Arguments

`beta1`	Tidy beta dataframe with columns `token`, `topic`, and `beta`, as returned by `tidy(model, matrix = 'beta')`.
`beta2`	Optional second tidy beta dataframe in the same format. If `NULL` (default), pairwise distances among the topics in `beta1` are returned.
`vocab`	Character vector of vocabulary tokens used to align the column space of both matrices. Tokens in `beta1` or `beta2` that are not in `vocab` are dropped; tokens in `vocab` absent from a beta are filled with probability 0.

Value

Numeric matrix of Hellinger distances. Dimensions are k1 × k1 when `beta2 = NULL`, or k1 × k2 when two beta dataframes are supplied, where k1 and k2 are the number of topics in each model.

Examples

```
set.seed(42)
vocab = letters[1:5]
make_beta = function(k) {
  rdirichlet(k, rep(1, length(vocab))) |>
    tibble::as_tibble(.name_repair = ~vocab) |>
    dplyr::mutate(topic = paste0('t', dplyr::row_number())) |>
    tidyr::pivot_longer(-topic, names_to = 'token', values_to = 'beta')
}
beta1 = make_beta(3)
beta2 = make_beta(4)
compare_betas(beta1, vocab = vocab)
compare_betas(beta1, beta2, vocab = vocab)
```

`draw_corpus` *Draw a collection of documents*

Description

Draw a collection of documents

Usage

```
draw_corpus(N, theta, phi)
```

Arguments

`N`	Length of documents
`theta`	Topic distribution for all documents, $n \times k$ matrix
`phi`	Word distribution for all topics, $k \times v$ matrix

Details

Standard pattern for generating a simulated DTM suitable for `tmfast()`:

```
set.seed(42)
theta  = rdirichlet(n_docs,   alpha = 1,   k = n_topics)
phi    = rdirichlet(n_topics, alpha = 0.1, k = vocab_size)
corpus = draw_corpus(rep(doc_length, n_docs), theta, phi)
model  = tmfast(corpus, n = n_topics)
```

`alpha = 1` for `theta` gives uniform topic mixing; `alpha = 0.1` for `phi` gives sparse, topic-specific word distributions. `doc_length` should be large enough that the full vocabulary is likely to appear (50–200 words per document is typical for a small simulated example).

Value

Document-term matrix, as a tibble, with columns `doc`, `word`, and `n`

See Also

Other generators: `journal_specific()`, `peak_alpha()`, `rdirichlet()`

Examples

```
set.seed(42)
theta  = rdirichlet(30, 1, k = 3)
phi    = rdirichlet(3, 0.1, k = 20)
corpus = draw_corpus(rep(50L, 30), theta, phi)
head(corpus)
```

entropy *Entropy of a distribution*

Description

Entropy of a distribution

Usage

```
entropy(p, base = 2)
```

Arguments

p	Discrete probability distribution
base	Desired base for entropy, eg, 2 for bits

Value

Calculated Shannon entropy

Examples

```
entropy(c(0.5, 0.5))
entropy(c(0.9, 0.1))
```

expected_entropy *Expected entropy for samples from a Dirichlet distribution*

Description

Samples P = <p1, p2, ..., pk> from Dirichlet distribution with parameter alpha = <alpha1, alpha2, ..., alphak> can be treated as categorical probability distributions with entropy $H(P) = \sum(-p \log p)$. This function calculates the expected entropy $E[H(P)]$ given alpha.

Usage

```
expected_entropy(alpha, k = NULL)
```

Arguments

alpha	Dirichlet parameter
k	If length(alpha) is 1, number of components in symmetric Dirichlet distribution

Details

After https://math.stackexchange.com/questions/2266285/expected-entropy-based-on-dirichlet-distribut
3195376#3195376

Value

Expected entropy $E[H(P)]$ in bits (log2 scale)

Examples

```
alpha = peak_alpha(50, 1)
set.seed(1357)
rdirichlet(500, alpha) |>
  apply(1, entropy) |>
  mean()
expected_entropy(alpha)
```

---

fit_varimax *Given a (rank* n*) PCA fit, return a rank* k < n *varimax fit*

Description

Given a (rank n) PCA fit, return a rank k < n varimax fit

Usage

```
fit_varimax(
  k,
  pca,
  feature_names,
  obs_names,
  varimax_fn = stats::varimax,
  varimax_opts = NULL,
  positive_skew = TRUE,
  x = NULL
)
```

Arguments

k	Desired rank of the fitted varimax model
pca	Fitted PCA model or varimaxes object
feature_names	Names of the features (eg, data columns)
obs_names	Names of the observations (eg, data rows)
varimax_fn	Function to use for varimax rotation
varimax_opts	Options passed to varimax_fn
positive_skew	Should negative-skewed factors be flipped to have positive skew?
x	PCA scores matrix (n_obs x max_k), as returned by predict(fitted, newdata). Used when the fitted pca object does not contain scores.

Details

After the initial rotation, factors with negative skew (left tails) are flipped pca must contain `$rotation` (feature loadings matrix) and `$sdev` (standard deviations per PC); `$x` (PC scores matrix) is also required unless `x` is supplied directly.

Value

List with components - `loadings`: Rotated feature loadings - `rotmat`: Rotation matrix - `scores`: Rotated observation scores

Examples

```
set.seed(42)
theta  = rdirichlet(50, 1, k = 3)
phi    = rdirichlet(3, 0.1, k = 20)
corpus = draw_corpus(rep(50L, 50), theta, phi)
dtm    = tidytext::cast_sparse(corpus, doc, word, n)
pca    = irlba::prcomp_irlba(dtm, n = 5)
fit_varimax(k = 3, pca = pca,
            feature_names = colnames(dtm),
            obs_names = rownames(dtm))
```

---

hellinger *Hellinger distances*

Description

Calculates Hellinger distance between rows of one or two matrices or tidied topic model dataframes.

Usage

```
hellinger(topics1, ...)

## S3 method for class 'Matrix'
hellinger(topics1, topics2 = NULL, ...)

## S3 method for class 'matrix'
hellinger(...)

## S3 method for class 'data.frame'
hellinger(
  topics1,
  id1 = "document",
  cat1 = "topic",
  prob1 = "prob",
  topics2 = NULL,
  id2 = "document",
```

```
  cat2 = "topic",
  prob2 = "prob",
  df = FALSE,
  ...
)
```

Arguments

topics1	First matrix ($n_1 \times k$), base R matrix, or tidied topic model dataframe.
...	Not used; required for S3 method compatibility.
topics2	Optional second matrix ($n_2 \times k$) or dataframe of the same type as topics1. When NULL (default), pairwise distances within topics1 are returned.
id1	Unit identifier column in topics1 (data.frame method only).
cat1	Category identifier column in topics1 (data.frame method only).
prob1	Probability value column in topics1 (data.frame method only).
id2	Unit identifier column in topics2 (data.frame method only).
cat2	Category identifier column in topics2 (data.frame method only).
prob2	Probability value column in topics2 (data.frame method only).
df	Should the function return the matrix of Hellinger distances (default) or a tidy dataframe? (data.frame method only)

Value

Matrix of size $n_1 \times n_1$ or $n_1 \times n_2$ (Matrix/matrix methods), or a matrix or tidy dataframe of Hellinger distances (data.frame method).

Examples

```
# Matrix / matrix method
set.seed(2022-06-09)
topics1 = rdirichlet(3, rep(5, 5))
topics2 = rdirichlet(3, rep(5, 5))
hellinger(topics1)
hellinger(topics1, topics2)

# data.frame method
set.seed(2022-06-09)
topics1 = rdirichlet(3, rep(5, 5)) |>
    tibble::as_tibble(rownames = 'doc_id') |>
    dplyr::mutate(doc_id = stringr::str_c('doc_', doc_id)) |>
    tidyr::pivot_longer(-doc_id,
                        names_to = 'topic',
                        values_to = 'gamma')
topics2 = rdirichlet(3, rep(5, 5)) |>
    tibble::as_tibble(rownames = 'doc_id') |>
    dplyr::mutate(doc_id = stringr::str_c('doc_', as.integer(doc_id) + 5)) |>
    tidyr::pivot_longer(-doc_id,
                        names_to = 'topic',
```

```
                        values_to = 'gamma')
hellinger(topics1, doc_id, prob1 = 'gamma', df = TRUE)
hellinger(topics1, doc_id, prob1 = 'gamma',
          topics2 = topics2, id2 = doc_id, prob2 = 'gamma')
```

`insert_topics` *Insert a topic model into a fitted* `tmfast`

Description

Apply varimax rotation for a value of k less than the maximum already included in the tmfast.

Usage

```
insert_topics(fitted, k, x = NULL)
```

Arguments

`fitted`	Fitted `tmfast` object
`k`	Desired number of topics for new model
`x`	Data matrix (document-term matrix), as Matrix object (eg, using `build_matrix()`)

Value

`tmfast` object, as `fitted`, with additional topic model inserted

Examples

```
set.seed(42)
theta  = rdirichlet(50, 1, k = 4)
phi    = rdirichlet(4, 0.1, k = 20)
corpus = draw_corpus(rep(50L, 50), theta, phi)
model  = tmfast(corpus, n = c(3, 4))
insert_topics(model, k = 2)
```

journal_specific *"Journal-specific" simulation scenario*

Description

Generates a corpus with `Mj` documents from `k` journals, each of which has a characteristic topic. Fits a varimax topic model of rank `k`, rotates the word-topic distribution to align with the true values, and reports Hellinger distance comparisons for each topic (word-topic) and document (topic-doc).

Usage

```
journal_specific(
  k = 5,
  Mj = 100,
  topic_peak = 0.8,
  topic_scale = 10,
  word_beta = 0.01,
  vocab = 10 * Mj * k,
  size = 3,
  mu = 300,
  bigjournal = FALSE,
  verbose = TRUE
)
```

Arguments

`k`	Number of topics/journals
`Mj`	Number of documents from each journal
`topic_peak`	Peak value for the asymmetric Dirichlet prior for true topic-doc distributions
`topic_scale`	Scale for the asymmetric Dirichlet prior for true topic-doc distributions
`word_beta`	Parameter for the symmetric Dirichlet prior for true word-doc distributions
`vocab`	Size of the vocabulary
`size`	Size parameter for the negative binomial distribution of document lengths
`mu`	Mean parameter for the negative binomial distribution of document lengths
`bigjournal`	Should the first journal have documents 10x as long (on average) as the others?
`verbose`	When TRUE, sends messages about the progress of the simulation

Value

A one-row `tibble::tibble()` with columns:

phi Mean Hellinger distance between true and fitted word-topic distributions

phi_vec List-column of per-topic Hellinger distances

theta Mean Hellinger distance between true and fitted document-topic distributions

theta_vec List-column of per-document Hellinger distances

See Also

Other generators: `draw_corpus()`, `peak_alpha()`, `rdirichlet()`

Examples

```
journal_specific(k = 2, Mj = 10, vocab = 50, verbose = FALSE)
```

`loadings` *Extract a PCA/varimax loadings matrix*

Description

Extract a PCA/varimax loadings matrix

Usage

```
loadings(x, ...)

## Default S3 method:
loadings(x, ...)
```

Arguments

`x`	Object to dispatch on
`...`	Passed to methods

Value

An object of class `"loadings"` (from **stats**), structured as a matrix with vocabulary terms as rows and varimax factors as columns. Values are the loading (weight) of each term on each factor.

Examples

```
set.seed(42)
theta  = rdirichlet(50, 1, k = 3)
phi    = rdirichlet(3, 0.1, k = 20)
corpus = draw_corpus(rep(50L, 50), theta, phi)
model  = tmfast(corpus, n = 3)
loadings(model, k = 3)

v = stats::varimax(matrix(runif(20), nrow = 5))
loadings(v)
```

ndH *Information gain (uniform distribution)*

Description

Calculates $\log_2 n \times \delta H$, the log total occurrence times information gain (relative to the uniform distribution) for each term. I prefer this for vocabulary selection over methods such as TF-IDF.

Usage

```
ndH(dataf, doc_col, term_col, count_col)
```

Arguments

dataf	Tidy document-term matrix
doc_col	Column of dataf with document IDs
term_col	Column of dataf with terms
count_col	Column of dataf with document-term counts

Value

Dataframe with columns

```
- `{{ term col }}`, term
- `dH`, information gain relative to uniform distribution over documents
- `n`, total count of term occurrence
- `ndH`, \eqn{\log_2 n \times \delta H}
```

Examples

```
library(dplyr)
library(tidytext)
library(janeaustenr)
austen_df = austen_books() |>
    unnest_tokens(term, text, token = 'words') |>
    mutate(author = 'Jane Austen') |>
    count(author, book, term)
ndH(austen_df, book, term, n)
```

ndR *Information gain (length-proportional distribution)*

Description

An alternative to `ndH()` that uses information gain relative to a distribution of documents that is proportional to length. With the uniform distribution and dramatic differences in document lengths (eg, over a few orders of magnitude), high-ndH terms tend to be distinctive terms from very long documents. With the length-proportional distribution, high information-gain terms are more likely to come from shorter documents. Informal testing suggests this approach performs better than the `ndH()` uniform distribution when documents have widely varying lengths, eg, over a few orders of magnitude.

Usage

```
ndR(dataf, doc_col, term_col, count_col)
```

Arguments

`dataf`	Tidy document-term matrix
`doc_col`	Column of `dataf` with document IDs
`term_col`	Column of `dataf` with terms
`count_col`	Column of `dataf` with document-term counts

Value

Dataframe with columns

```
- `{{ term col }}`, term
- `n`, total count of term occurrence
- `dR`, information gain relative to length-proportional distribution over documents
- `ndR`, \eqn{\log_2 n \times \delta R}
```

Examples

```
library(dplyr)
library(tidytext)
library(janeaustenr)
austen_df = austen_books() |>
    unnest_tokens(term, text, token = 'words') |>
    mutate(author = 'Jane Austen') |>
    count(author, book, term)
ndR(austen_df, book, term, n)
```

peak_alpha *Alpha parameter with a single peak*

Description

This function allows us to quickly define an `alpha` parameter for a Dirichlet distribution with a single (presumably high) `peak*scale` value at component `i` and all other components a uniform (presumably low) value `(1-peak)/(k-1)*scale`.

Usage

```
peak_alpha(k, i, peak = 0.8, scale = 1)
```

Arguments

`k`	Number of components
`i`	Index for the component that takes value `peak`
`peak`	Value for the single peak component
`scale`	Scaling factor applied to all concentration parameters

Value

Vector of length `k`

See Also

Other generators: `draw_corpus()`, `journal_specific()`, `rdirichlet()`

Examples

```
peak_alpha(5, 2)
peak_alpha(5, 2, peak = 0.9, scale = 10)
```

predict.varimaxes *Project new data into PCA score space*

Description

Project new data into PCA score space

Usage

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

Arguments

object	Fitted varimaxes or tmfast object
newdata	Document-term matrix (observations x terms) to project
...	Not used; included for S3 method compatibility.

Details

Projects newdata through the PCA rotation stored in object, returning raw PCA scores (not varimax scores). Intended for use in pipelines that combine new data with an existing fitted model (e.g., insert_topics()). Fragile: newdata must share the vocabulary of the training DTM, and the centering/scaling stored in object must match how the training data was prepared.

Memory warning: scale() coerces sparse matrices to dense. For large DTMs, this can be a substantial memory hazard. This mirrors the behavior of prcomp_irlba itself, which is why PCA scores are computed once at fit time and not re-projected on demand.

Value

Matrix of PCA scores (n_obs x max_k)

Examples

```
set.seed(42)
theta   = rdirichlet(50, 1, k = 3)
phi     = rdirichlet(3, 0.1, k = 20)
corpus  = draw_corpus(rep(50L, 50), theta, phi)
model   = tmfast(corpus, n = 3)
theta2  = rdirichlet(5, 1, k = 3)
newdocs = draw_corpus(rep(200L, 5), theta2, phi) |>
    tidytext::cast_sparse(doc, word, n)
predict(model, newdocs)
```

---

rdirichlet *Sample from the Dirichlet distribution*

Description

Sample from the Dirichlet distribution

Usage

```
rdirichlet(n, alpha, k = NULL)
```

Arguments

n	Number of samples (rows) to draw
alpha	Concentration parameters; either length 1 or length > 1 If length 1, assumes symmetric Dirichlet; k must not be null
k	Number of components (columns); ignored if length(alpha) > 1

Value

A matrix of `n` rows and `length(alpha)` or `k` columns

See Also

Other generators: `draw_corpus()`, `journal_specific()`, `peak_alpha()`

Examples

```
rdirichlet(10, .1, 5)
rdirichlet(10, c(.8, .1, .1))
```

`renorm` *Renormalize tidied distributions*

Description

Given a tidied dataframe of topic-doc or word-topic distributions and a exponent, renormalizes the distributions.

Usage

```
renorm(tidy_df, group_col, p_col, exponent, keep_original = FALSE)
```

Arguments

`tidy_df`	The tidied distribution dataframe
`group_col`	Grouping column, RHS of the conditional probability distribution, eg, topics for word-topic distributions
`p_col`	Column containing the probability for each category (eg, word) conditional on the group (eg, topic)
`exponent`	Exponent to use in renormalization
`keep_original`	Keep original probabilities?

Value

A dataframe with (if `keep_original` is `TRUE`) an added column of the form `p_col_rn` containing the renormalized probabilities or (if `keep_original` is `FALSE`) renormalized values in `p_col`.

Examples

```
set.seed(42)
theta  = rdirichlet(50, 1, k = 3)
phi    = rdirichlet(3, 0.1, k = 20)
corpus = draw_corpus(rep(50L, 50), theta, phi)
model  = tmfast(corpus, n = 3)
beta   = tidy(model, matrix = 'beta', k = 3)
pwr    = target_power(beta, topic, beta, target_entropy = 2)
renorm(beta, topic, beta, exponent = pwr)
```

---

rotation *Extract varimax rotation*

Description

Extract varimax rotation

Usage

```
rotation(x, ...)
```

Arguments

x	Object to dispatch on
...	Passed to methods

Value

A numeric k x k orthogonal rotation matrix, where k is the number of requested factors. This is the varimax rotation matrix used to transform PCA loadings into the rotated factor solution.

Examples

```
set.seed(42)
theta  = rdirichlet(50, 1, k = 3)
phi    = rdirichlet(3, 0.1, k = 20)
corpus = draw_corpus(rep(50L, 50), theta, phi)
model  = tmfast(corpus, n = 3)
rotation(model, k = 3)
```

scores *Extract item scores from a fitted PCA/varimax model*

Description

Extract item scores from a fitted PCA/varimax model

Usage

```
scores(x, ...)
```

Arguments

x	Object to dispatch on
...	Passed to methods

Value

A numeric matrix with documents as rows and varimax factors as columns. Values are the factor score for each document on each factor.

Examples

```
set.seed(42)
theta  = rdirichlet(50, 1, k = 3)
phi    = rdirichlet(3, 0.1, k = 20)
corpus = draw_corpus(rep(50L, 50), theta, phi)
model  = tmfast(corpus, n = 3)
scores(model, k = 3)
```

solve_power *Solve the equation to find the desired exponent*

Description

After https://stats.stackexchange.com/questions/521582/controlling-the-entropy-of-a-distribution

Usage

```
solve_power(p, target_H, return_full = FALSE)
```

Arguments

p	Initial distribution
target_H	Desired entropy for the transformed distribution
return_full	Return the full uniroot() output?

Value

Numeric value of the desired exponent

Examples

```
p = c(0.5, 0.3, 0.2)
solve_power(p, target_H = 1.0)
```

`target_power` *Find target power for renormalization*

Description

Given a tidied dataframe of topic-doc or word-topic distributions and a target entropy, find the mean exponent needed to adjust the temperature of each distribution to approximately match the target entropy.

Usage

```
target_power(tidy_df, group_col, p_col, target_entropy)
```

Arguments

`tidy_df`	The tidied distribution dataframe
`group_col`	Grouping column, RHS of the conditional probability distribution, eg, topics for word-topic distributions
`p_col`	Column containing the probability for each category (eg, word) conditional on the group (eg, topic)
`target_entropy`	Target entropy

Value

Mean exponent to renormalize to the target entropy

Examples

```
set.seed(42)
theta  = rdirichlet(50, 1, k = 3)
phi    = rdirichlet(3, 0.1, k = 20)
corpus = draw_corpus(rep(50L, 50), theta, phi)
model  = tmfast(corpus, n = 3)
beta   = tidy(model, matrix = 'beta', k = 3)
target_power(beta, topic, beta, target_entropy = 2)
```

`tidy.tmfast` *Extract beta and gamma matrices from* `tmfast` *objects*

Description

Extract beta and gamma matrices from `tmfast` objects

Usage

```
## S3 method for class 'tmfast'
tidy(
  x,
  k,
  matrix = "beta",
  df = TRUE,
  exponent = NULL,
  keep_original = FALSE,
  rotation = NULL,
  ...
)
```

Arguments

`x`	`tmfast` object
`k`	Index (number of topics/factors)
`matrix`	Desired matrix, either word-topic (`beta`) or topic-doc distributions (`gamma`)
`df`	Return a long dataframe (default) or wide matrix?
`exponent`	Renormalize the probabilities using a given exponent Applies only for `df == TRUE`
`keep_original`	If renormalizing, return original (pre-renormalized) probabilities?
`rotation`	Optional rotation matrix; see details
`...`	Not used; required for S3 method compatibility

Details

If `rotation` is not `NULL`, loadings/scores will be rotated. This might be used to align the fitted topics with known true topics, as in the `journal_specific` simulation. Loadings are left-multiplied by the given rotation, while scores are right-multiplied by the transpose of the given rotation.

Value

A long dataframe, with one row per word-topic or topic-doc combination. Column names depend on the value of `matrix`.

Examples

```
set.seed(42)
theta  = rdirichlet(50, 1, k = 3)
phi    = rdirichlet(3, 0.1, k = 20)
corpus = draw_corpus(rep(50L, 50), theta, phi)
model  = tmfast(corpus, n = 3)
tidy(model, k = 3, matrix = 'beta')
tidy(model, k = 3, matrix = 'gamma')
```

`tidy_all` *Extract gamma or beta matrices for all topics*

Description

Extract gamma or beta matrices for all topics

Usage

```
tidy_all(x, matrix = "beta", ...)
```

Arguments

`x`	`tmfast` object
`matrix`	Desired matrix, `'beta'` or `'gamma'`
`...`	Other arguments, passed to `tidy.tmfast()`

Value

A long dataframe, with one row per word-topic or topic-doc combination. Column names depend on the value of `matrix`.

Examples

```
set.seed(42)
theta  = rdirichlet(50, 1, k = 4)
phi    = rdirichlet(4, 0.1, k = 20)
corpus = draw_corpus(rep(50L, 50), theta, phi)
model  = tmfast(corpus, n = c(3, 4))
tidy_all(model, matrix = 'beta')
```

tmfast *Fit a topic model using PCA+varimax*

Description

Fit a topic model using PCA+varimax

Usage

```
tmfast(dtm, n, row = "doc", column = "word", value = "n", verbose = FALSE, ...)
```

Arguments

dtm	Document-term matrix. Either an object inheriting from Matrix or a long dataframe representation with row column row, column column column, and value column n.
n	Number of topics to return
row	In dataframe dtm, row column
column	In dataframe dtm, column column
value	In dataframe dtm, value column
verbose	Should irlba() be verbose?
...	Other arguments, passed to varimax_irlba

Details

If dtm is not a matrix, will be cast to a sparse matrix using tidytext::case_sparse()

Value

As per varimax_irlba, of class tmfast

tsne *Discursive space using t-SNE*

Description

2-dimensional "discursive space" representation of relationships between documents using Hellinger distances and t-SNE.

Usage

```
tsne(x, ...)

## S3 method for class 'data.frame'
tsne(x, doc_ids, perplexity = NULL, df = TRUE, ...)

## S3 method for class 'tmfast'
tsne(x, k, perplexity = NULL, df = TRUE, ...)

## S3 method for class 'STM'
tsne(x, doc_ids, perplexity = NULL, df = TRUE, ...)
```

Arguments

x	Fitted topic model (tmfast or STM)
...	Passed to methods
doc_ids	Vector of document IDs, in the same order as rows in x
perplexity	Perplexity parameter for t-SNE. By default, minimum of 30 and floor((length(doc_ids) - 1)/3) - 1.
df	Return a dataframe with columns document, x, and y (default) or the raw output of Rtsne.
k	Number of topics

Details

Algorithm checks distances to 3*perplexity nearest neighbors. Rtsne loses rownames (document IDs); these are either extracted from the tmfast object or passed separately for an STM object. Use set.seed() before calling for reproducibility.

Value

See df

Methods (by class)

- tsne(data.frame): Method for tidied gamma dataframes
- tsne(tmfast): Method for fitted tmfast objects
- tsne(STM): Method for fitted STM objects

Examples

```
set.seed(42)
theta = rdirichlet(50, 1, k = 3)
phi   = rdirichlet(3, 0.1, k = 30)
corpus = draw_corpus(rep(50L, 50), theta, phi)
fitted = tmfast(corpus, n = 3)
tsne(fitted, k = 3, df = TRUE)
```

umap *Discursive space using UMAP*

Description

2-dimensional "discursive space" representation of relationships between documents using Hellinger distances and UMAP.

Usage

```
umap(x, ...)

## S3 method for class 'matrix'
umap(x, include_data = FALSE, df = TRUE, ...)

## S3 method for class 'tmfast'
umap(x, k, ...)

## S3 method for class 'STM'
umap(x, doc_ids, ...)
```

Arguments

x	Fitted STM object
...	Passed to methods
include_data	Return the distance matrix inside the umap object? Default FALSE to save memory.
df	Return a tibble with columns document, x, and y (default) or the raw umap object.
k	Number of topics
doc_ids	Character vector of document IDs

Value

Tibble with columns document, x, y when df = TRUE; otherwise an object of class umap with components layout, knn, and config.

Methods (by class)

- umap(matrix): Method for distance matrices
- umap(tmfast): Method for fitted tmfast objects
- umap(STM): Method for fitted STM objects

Examples

```
gamma = rdirichlet(26, 1, 5)
rownames(gamma) = letters
h_gamma = hellinger(gamma)
umap(h_gamma, df = TRUE)

set.seed(42)
theta = rdirichlet(30, 1, k = 3)
phi   = rdirichlet(3, 0.1, k = 30)
corpus = draw_corpus(rep(50L, 30), theta, phi)
fitted = tmfast(corpus, n = 3)
umap(fitted, 3)
```

`varimax_irlba` *Fit a varimax-rotated PCA using irlba*

Description

Extract `n` principal components from the matrix `mx` using `irlba`, then rotate the solution using `varimax`

Usage

```
varimax_irlba(
  mx,
  n,
  prcomp_fn = irlba::prcomp_irlba,
  prcomp_opts = NULL,
  varimax_fn = stats::varimax,
  varimax_opts = NULL,
  retx = TRUE
)
```

Arguments

`mx`	Matrix of interest
`n`	Number of principal components / varimax factors to return; can take a vector of values
`prcomp_fn`	Function to use to extract principal components
`prcomp_opts`	List of options to pass to `prcomp_fn`
`varimax_fn`	Function to use for varimax rotation
`varimax_opts`	List of options to pass to `varimax_fn`
`retx`	Whether to return the input matrix `mx`

Value

A list of class `varimaxes`, with elements

- `totalvar`: Total variance, from PCA
- `sdev`: Standard deviations of the extracted principal components
- `x`: If `retx` is `TRUE`, the input matrix `mx`
- `rotation`: Rotation matrix (variable loadings) from PCA
- `varimaxes`: A list of class `varimaxes`, containing one fitted varimax model for each value of `n`, with further elements
  - `loadings`: Varimax-rotated standardized loadings
  - `rotmat`: Varimax rotation matrix
  - `scores`: Varimax-rotated observation scores

Examples

```
set.seed(42)
theta  = rdirichlet(50, 1, k = 3)
phi    = rdirichlet(3, 0.1, k = 20)
corpus = draw_corpus(rep(50L, 50), theta, phi)
dtm    = tidytext::cast_sparse(corpus, doc, word, n)
varimax_irlba(dtm, n = 3)
```

Index