Package 'fastTopics'

January 16, 2024

Encoding UTF-8

Type Package

Version 0.6-163

Date 2024-01-15

Title Fast Algorithms for Fitting Topic Models and Non-Negative Matrix Factorizations to Count Data

URL https://github.com/stephenslab/fastTopics

BugReports https://github.com/stephenslab/fastTopics/issues

Depends R (>= 3.3.0)

Description Implements fast, scalable optimization algorithms for fitting topic models (``grade of membership'' models) and non-negative matrix factorizations to count data. The methods exploit the special relationship between the multinomial topic model (also, ``probabilistic latent semantic indexing'') and Poisson non-negative matrix factorization. The package provides tools to compare, annotate and visualize model fits, including functions to efficiently create ``structure plots'' and identify key features in topics. The 'fastTopics' package is a successor to the 'CountClust' package. Note that the 'fastTopicis' package on GitHub has more vignettes illustrating application to single-cell RNA-seq data.

License BSD_2_clause + file LICENSE

Copyright inst/COPYRIGHTS

SystemRequirements GNU make

Imports graphics, utils, methods, stats, Matrix, gtools, quadprog, irlba, dplyr, Rtsne, uwot, ashr, Rcpp (>= 1.0.1), RcppParallel (>= 4.4.1), parallel, progress, pbapply, ggplot2 (>= 3.3.0), ggrepel (>= 0.9.0), cowplot, plotly, htmlwidgets

Suggests NNLM, Ternary, testthat, knitr, rmarkdown, RhpcBLASctl

LinkingTo Rcpp, RcppParallel, RcppArmadillo

Additional_repositories https://stephenslab.github.io/nnlm-drat

LazyData true

LazyDataCompression xz

NeedsCompilation yes

RoxygenNote 7.2.3

VignetteBuilder knitr

Author Peter Carbonetto [aut, cre],
Kevin Luo [aut],
Kushal Dey [aut],
Joyce Hsiao [ctb],
Abhishek Sarkar [ctb],
Anthony Hung [ctb],
Xihui Lin [ctb],
Paul C. Boutros [ctb],
Minzhe Wang [ctb],
Tracy Ke [ctb],
Matthew Stephens [aut]

Maintainer Peter Carbonetto <peter.carbonetto@gmail.com>

Repository CRAN

Date/Publication 2024-01-15 23:30:09 UTC

R topics documented:

`compare_fits` *Summarize and Compare Model Fits*

Description

Create a table summarizing the results of fitting one or more Poisson non-negative matrix factorizations or multinomial topic models.

Usage

```
compare_fits(fits)
```

Arguments

`fits`	An object of class `"poisson_nmf_fit"` or `"multinom_topic_model_fit"`, or a non-empty, named list in which all list elements are Poisson NMF model fits or all multinomial topic model fits.

Value

A data frame with one row per element of `fits`, and with the following columns:

`k`	The rank of the matrix factorization.
`loglik`	The log-likelihood (either Poisson NMF or multinomial topic model likelihood) achieved at the last model fitting update.
`dev`	For Poisson NMF model fits only, the deviance achieved at the last model fitting update.
`res`	The maximum residual of the Karush-Kuhn-Tucker (KKT) system achieved at the last model fitting update; small values indicate that the solution is close to a local maximum, or stationary point, of the likelihood.
`loglik.diff`	The improvement in the log-likelihood relative to the model fit with the smallest log-likelihood.
`dev.diff`	The improvement in the deviance relative to the model fit with the largest deviance.
`nonzeros.f`	The rate of nonzeros in the factors matrix, as determined by `control$zero.threshold`.
`nonzeros.l`	The rate of nonzeros in the loadings matrix, as determined by `control$zero.threshold`.
`numiter`	The number of loadings and/or factor updates performed.
`runtime`	The total runtime (in s) of the model fitting updates.

See Also

`fit_poisson_nmf`, `fit_topic_model`

de_analysis *Differential Expression Analysis using a Topic Model*

Description

Implements methods for differential expression analysis using a topic model. These methods are motivated by gene expression studies, but could have other uses, such as identifying "key words" for topics.

Usage

```
de_analysis(
  fit,
  X,
  s = rowSums(X),
  pseudocount = 0.01,
  fit.method = c("scd", "em", "mu", "ccd", "glm"),
  shrink.method = c("ash", "none"),
  lfc.stat = "le",
  control = list(),
  verbose = TRUE,
  ...
)

de_analysis_control_default()
```

Arguments

`fit`	An object of class "poisson_nmf_fit" or "multinom_topic_model_fit", or an n x k matrix of topic proportions, where k is the number of topics. (The elements in each row of this matrix should sum to 1.) If a Poisson NMF fit is provided as input, the corresponding multinomial topic model fit is automatically recovered using `poisson2multinom`.
`X`	The n x m counts matrix. It can be a sparse matrix (class `"dgCMatrix"`) or dense matrix (class `"matrix"`).
`s`	A numeric vector of length n determining how the rates are scaled in the Poisson models. See "Details" for guidance on the choice of `s`.
`pseudocount`	Observations with this value are added to the counts matrix to stabilize maximum-likelihood estimation.
`fit.method`	Method used to fit the Poisson models. Note that `fit.method = "glm"` is the slowest method, and is mainly used for testing.
`shrink.method`	Method used to stabilize the posterior mean LFC estimates. When `shrink.method = "ash"`, the "adaptive shrinkage" method implemented in the 'ashr' package is used to compute posterior. When `shrink.method = "none"`, no stabilization is performed, and the "raw" posterior mean LFC estimates are returned.

`lfc.stat`	The log-fold change statistics returned: `lfc.stat = "vsnull"`, the log-fold change relative to the null; `lfc.stat = "le"`, the "least extreme" LFC; or a topic name or number, in which case the LFC is defined relative to the selected topic. See "Details" for more detailed explanations of these choices.
`control`	A list of parameters controlling behaviour of the optimization and Monte Carlo algorithms. See 'Details'.
`verbose`	When `verbose = TRUE`, progress information is printed to the console.
`...`	When `shrink.method = "ash"`, these are additional arguments passed to ash.

Details

The methods are based on the Poisson model

$$x_i \; Poisson(\lambda_i),$$

in which the Poisson rates are

$$\lambda_i = \sum_{j=1}^{k} s_i l_{ij} f_j,$$

the l_{ik} are the topic proportions and the f_j are the unknowns to be estimated. This model is applied separately to each column of `X`. When s_i (specified by input argument `s`) is equal the total count in row i (this is the default), the Poisson model will closely approximate a binomial model of the count data, and the unknowns f_j will approximate binomial probabilities. (The Poisson approximation to the binomial is most accurate when the total counts `rowSums(X)` are large and the unknowns f_j are small.) Other choices for `s` are possible, and implement different normalization schemes.

To allow for some flexibility, `de_analysis` allows for the log-fold change to be measured in several ways.

One option is to compare against the probability under the null model: $LFC(j) = log2(f_j/f_0)$, where f_0 is the single parameter in the Poisson model $x_i \; Poisson(\lambda_i)$ with rates $\lambda_i = s_i f_0$. This LFC definition is chosen with `lfc.stat = "vsnull"`.

Another option is to compare against a chosen topic, k: $LFC(j) = log2(f_j/f_k)$. By definition, $LFC(k)$ is zero, and statistics such as z-scores and p-values for topic k are set to `NA`. This LFC definition is selected by setting `lfc.stat = k`.

A final option (which is the default) computes the "least extreme" LFC, defined as $LFC(j) = log2(f_j/f_k)$ such that k is the topic other than j that gives the ratio f_j/f_k closest to 1. This option is chosen with `lfc.stat = "le"`.

We recommend setting `shrink.method = "ash"`, which uses the "adaptive shrinkage" method (Stephens, 2016) to improve accuracy of the posterior mean estimates and z-scores. We follow the settings used in `lfcShrink` from the 'DESeq2' package, with `type = "ashr"`.

Note that all LFC statistics are defined using the base-2 logarithm following the conventioned used in differential expression analysis.

The `control` argument is a list in which any of the following named components will override the default optimization algorithm settings (as they are defined by `de_analysis_control_default`):

`numiter` Maximum number of iterations performed in fitting the Poisson models. When `fit.method = "glm"`, this is passed as argument `maxit` to the `glm` function.

`minval` A small, positive number. All topic proportions less than this value and greater than `1 - minval` are set to this value.

`tol` Controls the convergence tolerance for fitting the Poisson models. When `fit.method = "glm"`, this is passed as argument `epsilon` to function `glm`.

`conf.level` The size of the highest posterior density (HPD) intervals. Should be a number greater than 0 and less than 1.

`ns` Number of Monte Carlo samples simulated by random-walk MCMC for estimating posterior LFC quantities.

`rw` The standard deviation of the normal density used to propose new states in the random-walk MCMC.

`eps` A small, non-negative number added to the terms inside the logarithms to avoid computing logarithms of zero.

`nc` Number of threads used in the multithreaded computations. Note that the multithreading relies on forking hence is not available on Windows; will return an error on Windows unless `nc = 1`. See `mclapply` for details. Also note that if R is installed with a multithreading numerical linear algebra library (e.g., OpenBLAS), for best performance the number of threads used by the linear algebra library should be set to 1 (i.e., no multithreading). This can be controlled for example using the RhpcBLASctl package.

`nsplit` The number of data splits used in the multithreaded computations (only relevant when `nc > 1`). More splits increase the granularity of the progress bar, but can also slow down the mutithreaded computations by introducing more overhead in the call to `pblapply`.

Value

A list with the following elements:

`est`	The log-fold change maximum-likelihood estimates.
`postmean`	Posterior mean LFC estimates.
`lower`	Lower limits of estimated HPD intervals. Note that these are not updated by the shrinkage step.
`upper`	Upper limits of estimated HPD intervals. Note that these are not updated by the shrinkage step.
`z`	z-scores for posterior mean LFC estimates.
`lpval`	-log10 two-tailed p-values obtained from the z-scores. When `shrink.method = "ash"`, this is `NA`, and the s-values are returned instead (see below).
`svalue`	s-values returned by `ash`. s-values are analogous to q-values, but based on the local false sign rate; see Stephens (2016).
`lfsr`	When `shrink.method = "ash"` only, this output contains the estimated local false sign rates.
`ash`	When `shrink.method = "ash"` only, this output contains the `ash` return value (after removing the `"data"`, `"result"` and `"call"` list elements).
`F`	Maximum-likelihood estimates of the Poisson model parameters.
`f0`	Maximum-likelihood estimates of the null model parameters.
`ar`	A vector containing the Metropolis acceptance ratios from each MCMC run.

References

Stephens, M. (2016). False discovery rates: a new deal. *Biostatistics* **18**(2), kxw041. doi: 10.1093/biostatistics/kxw041

Zhu, A., Ibrahim, J. G. and Love, M. I. (2019). Heavy-tailed prior distributions for sequence count data: removing the noise and preserving large differences. *Bioinformatics* **35**(12), 2084–2092.

Examples

```
# Perform a differential expression (DE) analysis using the previously
# fitted multinomial topic model. Note that the de_analysis call could
# take several minutes to complete.

set.seed(1)
data(pbmc_facs)
de <- de_analysis(pbmc_facs$fit,pbmc_facs$counts)

# Compile the DE analysis results for topic 4 into a table, and
# rank the genes by the posterior mean log-fold change, limiting to
# DE genes identified with low lfsr ("local false sign rate").
k <- 4
dat <- data.frame(postmean = de$postmean[,k],
                  z        = de$z[,k],
                  lfsr     = de$lfsr[,k])
rownames(dat) <- with(pbmc_facs$genes,paste(symbol,ensembl,sep = "_"))
dat <- subset(dat,lfsr < 0.01)
dat <- dat[order(dat$postmean,decreasing = TRUE),]

# Genes at the top of this ranking are genes that are much more
# highly expressed in the topic compared to other topics.
head(dat,n = 10)

# The genes at the bottom of the ranking are genes that are much less
# expressed in the topic.
tail(dat,n = 10)

# Create a volcano plot from the DE results for topic 4.
volcano_plot(de,k = k,ymax = 50,labels = pbmc_facs$genes$symbol)
```

---

embedding_plot_2d *PCA, t-SNE and UMAP Plots*

Description

Visualize the structure of the Poisson NMF loadings or the multinomial topic model topic proportions by projection onto a 2-d surface. `plot_hexbin_plot` is most useful for visualizing the PCs of a data set with thousands of samples or more.

Usage

```
embedding_plot_2d(
  fit,
  Y,
  fill = "loading",
  k,
  fill.label,
  ggplot_call = embedding_plot_2d_ggplot_call,
  plot_grid_call = function(plots) do.call(plot_grid, plots)
)

embedding_plot_2d_ggplot_call(
  Y,
  fill,
  fill.type = c("loading", "numeric", "factor", "none"),
  fill.label,
  font.size = 9
)

pca_plot(
  fit,
  Y,
  pcs = 1:2,
  n = 10000,
  fill = "loading",
  k,
  fill.label,
  ggplot_call = embedding_plot_2d_ggplot_call,
  plot_grid_call = function(plots) do.call(plot_grid, plots),
  ...
)

tsne_plot(
  fit,
  Y,
  n = 2000,
  fill = "loading",
  k,
  fill.label,
  ggplot_call = embedding_plot_2d_ggplot_call,
  plot_grid_call = function(plots) do.call(plot_grid, plots),
  ...
)

umap_plot(
  fit,
  Y,
  n = 2000,
```

```
  fill = "loading",
  k,
  fill.label,
  ggplot_call = embedding_plot_2d_ggplot_call,
  plot_grid_call = function(plots) do.call(plot_grid, plots),
  ...
)

pca_hexbin_plot(
  fit,
  Y,
  pcs = 1:2,
  bins = 40,
  breaks = c(0, 1, 10, 100, 1000, Inf),
  ggplot_call = pca_hexbin_plot_ggplot_call,
  ...
)

pca_hexbin_plot_ggplot_call(Y, bins, breaks, font.size = 9)
```

Arguments

`fit`	An object of class "poisson_nmf_fit" or "multinom_topic_model_fit".
`Y`	The n x 2 matrix containing the 2-d embedding, where n is the number of rows in `fit$L`. If not provided, the embedding will be computed automatically.
`fill`	The quantity to map onto the fill colour of the points in the PCA plot. Set `fill = "loading"` to vary the fill colour according to the loadings (or topic proportions) of the select topiced or topics. Alternatively, `fill` may be set to a data vector with one entry per row of `fit$L`, in which case the data are mapped to the fill colour of the points. When `fill = "none"`, the fill colour is not varied.
`k`	The dimensions or topics selected by number or name. When `fill = "loading"`, one plot is created per selected dimension or topic; when `fill = "loading"` and `k` is not specified, all dimensions or topics are plotted.
`fill.label`	The label used for the fill colour legend.
`ggplot_call`	The function used to create the plot. Replace `embedding_plot_2d_ggplot_call` or `pca_hexbin_plot_ggplot_call` with your own function to customize the appearance of the plot.
`plot_grid_call`	When `fill = "loading"` and multiple topics (`k`) are selected, this is the function used to arrange the plots into a grid using `plot_grid`. It should be a function accepting a single argument, `plots`, a list of `ggplot` objects.
`fill.type`	The type of variable mapped to fill colour. The fill colour is not varied when `fill.type = "none"`.
`font.size`	Font size used in plot.
`pcs`	The two principal components (PCs) to be plotted, specified by name or number.
`n`	The maximum number of points to plot. If `n` is less than the number of rows of `fit$L`, the rows are subsampled at random. This argument is ignored if `Y` is provided.

...	Additional arguments passed to pca_from_topics, tsne_from_topics or umap_from_topics. These additional arguments are only used if Y is not provided.
bins	Number of bins used to create hexagonal 2-d histogram. Passed as the "bins" argument to stat_bin_hex.
breaks	To produce the hexagonal histogram, the counts are subdivided into intervals based on breaks. Passed as the "breaks" argument to cut.

Details

This is a lightweight interface primarily intended to expedite creation of plots for visualizing the loadings or topic proportions; most of the heavy lifting is done by 'ggplot2'. The 2-d embedding itself is computed by invoking pca_from_topics, tsne_from_topics or umap_from_topics. For more control over the plot's appearance, the plot can be customized by modifying the ggplot_call and plot_grid_call arguments.

An effective 2-d visualization may also require some fine-tunning of the settings, such as the t-SNE "perplexity", or the number of samples included in the plot. The PCA, UMAP, t-SNE settings can be controlled by the additional arguments (...). Alternatively, a 2-d embedding may be pre-computed, and passed as argument Y.

Value

A ggplot object.

See Also

pca_from_topics, tsne_from_topics, umap_from_topics

Examples

```
set.seed(1)
data(pbmc_facs)

# Get the Poisson NMF and multinomial topic models fitted to the
# PBMC data.
fit1 <- multinom2poisson(pbmc_facs$fit)
fit2 <- pbmc_facs$fit

# Plot the first two PCs of the loadings matrix (for the
# multinomial topic model, "fit2", the loadings are the topic
# proportions).
subpop <- pbmc_facs$samples$subpop
p1 <- pca_plot(fit1,k = 1)
p2 <- pca_plot(fit2)
p3 <- pca_plot(fit2,fill = "none")
p4 <- pca_plot(fit2,pcs = 3:4,fill = "none")
p5 <- pca_plot(fit2,fill = fit2$L[,1])
p6 <- pca_plot(fit2,fill = subpop)
p7 <- pca_hexbin_plot(fit1)
p8 <- pca_hexbin_plot(fit2)
```

```
# Plot the loadings using t-SNE.
p1 <- tsne_plot(fit1,k = 1)
p2 <- tsne_plot(fit2)
p3 <- tsne_plot(fit2,fill = subpop)

# Plot the loadings using UMAP.
p1 <- umap_plot(fit1,k = 1)
p2 <- umap_plot(fit2)
p3 <- umap_plot(fit2,fill = subpop)
```

`fit_multinom_model` *Fit Simple Multinomial Model*

Description

Fit a simple multinomial model for count data, in which each sample (*i.e.*, a row of the data matrix `X`) is assigned to a cluster. Under this simple multinomial model, x_{ij} assigned to cluster k is multinomial with sample size $s_i = x_{i1} + ... + x_{im}$ and multinomial probabilities $p_{1k}, ..., p_{mk}$. This is a special case of the multinomial topic model in which all the mixture proportions are either 0 or 1. The maximum-likelihood estimates (MLEs) of the multinomial probabilities have a closed-form solution; no iterative algorithm is needed to fit this simple model.

Usage

```
fit_multinom_model(cluster, X, verbose = c("none", "detailed"), ...)
```

Arguments

`cluster`	A factor specifying a grouping, or clustering, of the rows of `X`; e.g., the "cluster" output from `kmeans`.
`X`	The n x m matrix of counts; all entries of X should be non-negative. It can be a sparse matrix (class `"dgCMatrix"`) or dense matrix (class `"matrix"`), with some exceptions (see 'Details').
`verbose`	This is passed as the "verbose" argument in the call to `init_poisson_nmf`.
`...`	Additional arguments passed to `init_poisson_nmf`.

Value

A multinomial topic model fit.

See Also

`fit_topic_model`

`fit_poisson_nmf` *Fit Non-negative Matrix Factorization to Count Data*

Description

Approximate the input matrix `X` by the non-negative matrix factorization `tcrossprod(L,F)`, in which the quality of the approximation is measured by a "divergence" criterion; equivalently, optimize the likelihood under a Poisson model of the count data, `X`, in which the Poisson rates are given by `tcrossprod(L,F)`. Function `fit_poisson_nmf` runs a specified number of coordinate-wise updates to fit the L and F matrices.

Usage

```
fit_poisson_nmf(
  X,
  k,
  fit0,
  numiter = 100,
  update.factors = seq(1, ncol(X)),
  update.loadings = seq(1, nrow(X)),
  method = c("scd", "em", "mu", "ccd"),
  init.method = c("topicscore", "random"),
  control = list(),
  verbose = c("progressbar", "detailed", "none")
)

fit_poisson_nmf_control_default()

init_poisson_nmf(
  X,
  F,
  L,
  k,
  init.method = c("topicscore", "random"),
  beta = 0.5,
  betamax = 0.99,
  control = list(),
  verbose = c("detailed", "none")
)

init_poisson_nmf_from_clustering(X, clusters, ...)
```

Arguments

`X`	The n x m matrix of counts; all entries of X should be non-negative. It can be a sparse matrix (class `"dgCMatrix"`) or dense matrix (class `"matrix"`), with some exceptions (see 'Details').

k — An integer 2 or greater giving the matrix rank. This argument should only be specified if the initial fit (fit0 or F, L) is not provided.

fit0 — The initial model fit. It should be an object of class "poisson_nmf_fit", such as an output from init_poisson_nmf, or from a previous call to fit_poisson_nmf.

numiter — The number of updates of the factors and loadings to perform.

update.factors — A numeric vector specifying which factors (rows of F) to update. By default, all factors are updated. Note that the rows that are not updated may still change by rescaling. When NULL, all factors are fixed. This option is only implemented for method = "em" and method = "scd". If another method is selected, the default setting of update.factors must be used.

update.loadings — A numeric vector specifying which loadings (rows of L) to update. By default, all loadings are updated. Note that the rows that are not updated may still change by rescaling. When NULL, all loadings are fixed. This option is only implemented for method = "em" and method = "scd". If another method is selected, the default setting of update.loadings must be used.

method — The method to use for updating the factors and loadings. Four methods are implemented: multiplicative updates, method = "mu"; expectation maximization (EM), method = "em"; sequential co-ordinate descent (SCD), method = "scd"; and cyclic co-ordinate descent (CCD), method = "ccd". See 'Details' for a detailed description of these methods.

init.method — The method used to initialize the factors and loadings. When init.method = "random", the factors and loadings are initialized uniformly at random; when init.method = "topicscore", the factors are initialized using the (very fast) Topic SCORE algorithm (Ke & Wang, 2017), and the loadings are initialized by running a small number of SCD updates. This input argument is ignored if initial estimates of the factors and loadings are already provided via input fit0, or inputs F and L.

control — A list of parameters controlling the behaviour of the optimization algorithm (and the Topic SCORE algorithm if it is used to initialize the model parameters). See 'Details'.

verbose — When verbose = "detailed", information about the algorithm's progress is printed to the console at each iteration; when verbose = "progressbar", a progress bar is shown; and when verbose = "none", no progress information is printed. See the description of the "progress" return value for an explanation of verbose = "detailed" console output. (Note that some columns of the "progress" data frame are not shown in the console output.)

F — An optional argument giving is the initial estimate of the factors (also known as "basis vectors"). It should be an m x k matrix, where m is the number of columns in the counts matrix X, and k > 1 is the rank of the matrix factorization (equivalently, the number of "topics"). All entries of F should be non-negative. When F and L are not provided, input argument k should be specified instead.

L — An optional argument giving the initial estimate of the loadings (also known as "activations"). It should be an n x k matrix, where n is the number of rows in the counts matrix X, and k > 1 is the rank of the matrix factorization (equivalently,

	the number of "topics"). All entries of `L` should be non-negative. When `F` and `L` are not provided, input argument `k` should be specified instead.
`beta`	Initial setting of the extrapolation parameter. This is $beta$ in Algorithm 3 of Ang & Gillis (2019).
`betamax`	Initial setting for the upper bound on the extrapolation parameter. This is $\bar{\gamma}$ in Algorithm 3 of Ang & Gillis (2019).
`clusters`	A factor specifying a grouping, or clustering, of the rows of `X`.
`...`	Additional arguments passed to `init_poisson_nmf`.

Details

In Poisson non-negative matrix factorization (Lee & Seung, 2001), counts x_{ij} in the $n \times m$ matrix, X, are modeled by the Poisson distribution:

$$x_{ij} \sim \text{Poisson}(\lambda_{ij}).$$

Each Poisson rate, λ_{ij}, is a linear combination of parameters $f_{jk} \geq 0, l_{ik} \geq 0$ to be fitted to the data:

$$\lambda_{ij} = \sum_{k=1}^{K} l_{ik} f_{jk},$$

in which K is a user-specified tuning parameter specifying the rank of the matrix factorization. Function `fit_poisson_nmf` computes maximum-likelihood estimates (MLEs) of the parameters. For additional mathematical background, and an explanation of how Poisson NMF is connected to topic modeling, see the vignette: `vignette(topic = "relationship",package = "fastTopics")`.

Using this function requires some care; only minimal argument checking is performed, and error messages may not be helpful.

The EM and multiplicative updates are simple and fast, but can be slow to converge to a stationary point. When `control$numiter = 1`, the EM and multiplicative updates are mathematically equivalent to the multiplicative updates, and therefore share the same convergence properties. However, the implementation of the EM updates is quite different; in particular, the EM updates are more suitable for sparse counts matrices. The implementation of the multiplicative updates is adapted from the MATLAB code by Daichi Kitamura http://d-kitamura.net.

Since the multiplicative updates are implemented using standard matrix operations, the speed is heavily dependent on the BLAS/LAPACK numerical libraries used. In particular, using optimized implementations such as OpenBLAS or Intel MKL can result in much improved performance of the multiplcative updates.

The cyclic co-ordinate descent (CCD) and sequential co-ordinate descent (SCD) updates adopt the same optimization strategy, but differ in the implementation details. In practice, we have found that the CCD and SCD updates arrive at the same solution when initialized "sufficiently close" to a stationary point. The CCD implementation is adapted from the C++ code developed by Cho-Jui Hsieh and Inderjit Dhillon, which is available for download at https://www.cs.utexas.edu/~cjhsieh/nmf/. The SCD implementation is based on version 0.4-3 of the 'NNLM' package.

An additional re-scaling step is performed after each update to promote numerical stability.

We use three measures of progress for the model fitting: (1) improvement in the log-likelihood (or deviance), (2) change in the model parameters, and (3) the residuals of the Karush-Kuhn-Tucker

(KKT) first-order conditions. As the iterates approach a stationary point of the loss function, the change in the model parameters should be small, and the residuals of the KKT system should vanish. Use `plot_progress` to plot the improvement in the solution over time.

See `fit_topic_model` for additional guidance on model fitting, particularly for large or complex data sets.

The `control` argument is a list in which any of the following named components will override the default optimization algorithm settings (as they are defined by `fit_poisson_nmf_control_default`):

`numiter` Number of "inner loop" iterations to run when performing and update of the factors or loadings. This must be set to 1 for `method = "mu"` and `method = "ccd"`.

`nc` Number of RcppParallel threads to use for the updates. When `nc` is `NA`, the number of threads is determined by calling `defaultNumThreads`. This setting is ignored for the multiplicative upates (`method = "mu"`).

`minval` A small, positive constant used to safeguard the multiplicative updates. The safeguarded updates are implemented as `F <- pmax(F1,minval)` and `L <- pmax(L1,minval)`, where `F1` and `L1` are the factors and loadings matrices obtained by applying an update. This is motivated by Theorem 1 of Gillis & Glineur (2012). Setting `minval = 0` is allowed, but some methods are not guaranteed to converge to a stationary point without this safeguard, and a warning will be given in this case.

`extrapolate` When `extrapolate = TRUE`, the extrapolation scheme of Ang & Gillis (2019) is used.

`extrapolate.reset` To promote better numerical stability of the extrapolated updates, they are "reset" every so often. This parameter determines the number of iterations to wait before resetting.

`beta.increase` When the extrapolated update improves the solution, scale the extrapolation parameter by this amount.

`beta.reduce` When the extrapolaaed update does not improve the solution, scale the extrapolation parameter by this amount.

`betamax.increase` When the extrapolated update improves the solution, scale the extrapolation parameter by this amount.

`eps` A small, non-negative number that is added to the terms inside the logarithms to sidestep computing logarithms of zero. This prevents numerical problems at the cost of introducing a small inaccuracy in the solution. Increasing this number may lead to faster convergence but possibly a less accurate solution.

`zero.threshold` A small, non-negative number used to determine which entries of the solution are exactly zero. Any entries that are less than or equal to `zero.threshold` are considered to be exactly zero.

An additional setting, `control$init.numiter`, controls the number of sequential co-ordinate descent (SCD) updates that are performed to initialize the loadings matrix when `init.method = "topicscore"`.

Value

`init_poisson_nmf` and `fit_poisson_nmf` both return an object capturing the optimization algorithm state (for `init_poisson_nmf`, this is the initial state). It is a list with the following elements:

F	A matrix containing the current best estimates of the factors.
L	A matrix containing the current best estimates of the loadings.
Fn	A matrix containing the non-extrapolated factor estimates. If extrapolation is not used, Fn and F will be the same.
Ln	A matrix containing the non-extrapolated estimates of the loadings. If extrapolation is not used, Ln and L will be the same.
Fy	A matrix containing the extrapolated factor estimates. If the extrapolation scheme is not used, Fy and F will be the same.
Ly	A matrix containing the extrapolated estimates of the loadings. If extrapolation is not used, Ly and L will be the same.
loss	Value of the objective ("loss") function computed at the current best estimates of the factors and loadings.
loss.fnly	Value of the objective ("loss") function computed at the extrapolated solution for the loadings (Ly) and the non-extrapolated solution for the factors (Fn). This is used internally to implement the extrapolated updates.
iter	The number of the most recently completed iteration.
beta	The extrapolation parameter, $beta$ in Algorithm 3 of Ang & Gillis (2019).
betamax	Upper bound on the extrapolation parameter. This is $\bar{\gamma}$ in Algorithm 3 of Ang & Gillis (2019).
beta0	The setting of the extrapolation parameter at the last iteration that improved the solution.
progress	A data frame containing detailed information about the algorithm's progress. The data frame should have numiter rows. The columns of the data frame are: "iter", the iteration number; "loglik", the Poisson NMF log-likelihood at the current best factor and loading estimates; "loglik.multinom", the multinomial topic model log-likelihood at the current best factor and loading estimates; "dev", the deviance at the current best factor and loading estimates; "res", the maximum residual of the Karush-Kuhn-Tucker (KKT) first-order optimality conditions at the current best factor and loading estimates; "delta.f", the largest change in the factors matrix; "delta.l", the largest change in the loadings matrix; "nonzeros.f", the proportion of entries in the factors matrix that are nonzero; "nonzeros.l", the proportion of entries in the loadings matrix that are nonzero; "extrapolate", which is 1 if extrapolation is used, otherwise it is 0; "beta", the setting of the extrapolation parameter; "betamax", the setting of the extrapolation parameter upper bound; and "timing", the elapsed time in seconds (recorded using proc.time).

References

Ang, A. and Gillis, N. (2019). Accelerating nonnegative matrix factorization algorithms using extrapolation. *Neural Computation* **31**, 417–439.

Cichocki, A., Cruces, S. and Amari, S. (2011). Generalized alpha-beta divergences and their application to robust nonnegative matrix factorization. *Entropy* **13**, 134–170.

Gillis, N. and Glineur, F. (2012). Accelerated multiplicative updates and hierarchical ALS algorithms for nonnegative matrix factorization. *Neural Computation* 24, 1085–1105.

Hsieh, C.-J. and Dhillon, I. (2011). Fast coordinate descent methods with variable selection for non-negative matrix factorization. In *Proceedings of the 17th ACM SIGKDD international conference on Knowledge discovery and data mining*, p. 1064-1072

Lee, D. D. and Seung, H. S. (2001). Algorithms for non-negative matrix factorization. In *Advances in Neural Information Processing Systems* **13**, 556–562.

Lin, X. and Boutros, P. C. (2018). Optimization and expansion of non-negative matrix factorization. *BMC Bioinformatics* **21**, 7.

Ke, Z. & Wang, M. (2017). A new SVD approach to optimal topic estimation. *arXiv* https://arxiv.org/abs/1704.07016

See Also

fit_topic_model, plot_progress

Examples

```
# Simulate a (sparse) 80 x 100 counts matrix.
library(Matrix)
set.seed(1)
X <- simulate_count_data(80,100,k = 3,sparse = TRUE)$X

# Remove columns (words) that do not appear in any row (document).
X <- X[,colSums(X > 0) > 0]

# Run 10 EM updates to find a good initialization.
fit0 <- fit_poisson_nmf(X,k = 3,numiter = 10,method = "em")

# Fit the Poisson NMF model by running 50 EM updates.
fit_em <- fit_poisson_nmf(X,fit0 = fit0,numiter = 50,method = "em")

# Fit the Poisson NMF model by running 50 extrapolated SCD updates.
fit_scd <- fit_poisson_nmf(X,fit0 = fit0,numiter = 50,method = "scd",
                           control = list(extrapolate = TRUE))

# Compare the two fits.
fits <- list(em = fit_em,scd = fit_scd)
compare_fits(fits)
plot_progress(fits,y = "loglik")
plot_progress(fits,y = "res")

# Recover the topic model. After this step, the L matrix contains the
# mixture proportions ("loadings"), and the F matrix contains the
# word frequencies ("factors").
fit_multinom <- poisson2multinom(fit_scd)
```

`fit_topic_model` *Simple Interface for Fitting a Multinomial Topic Model*

Description

Fits a multinomial topic model to the count data, hiding most of the complexities of model fitting. The default optimization settings used here are intended to work well in a wide range of data sets, although some fine-tuning may be needed for more difficult cases. For full control, use `fit_poisson_nmf`.

Usage

```
fit_topic_model(
  X,
  k,
  numiter.main = 100,
  numiter.refine = 100,
  method.main = "em",
  method.refine = "scd",
  init.method = c("topicscore", "random"),
  control.init = list(),
  control.main = list(numiter = 4),
  control.refine = list(numiter = 4, extrapolate = TRUE),
  verbose = c("progressbar", "detailed", "none")
)
```

Arguments

`X`	The n x m matrix of counts; all entries of X should be non-negative. It can be a sparse matrix (class `"dgCMatrix"`) or dense matrix (class `"matrix"`).
`k`	The number of topics. Must be 2 or greater.
`numiter.main`	Number of updates of the factors and loadings to perform in the main model fitting step. Should be 1 or more.
`numiter.refine`	Number of updates of the factors and loadings to perform in the model refinement step.
`method.main`	The method to use for updating the factors and loadings in the main model fitting step. Passed as argument "method" to `fit_poisson_nmf`.
`method.refine`	The method to use for updating the factors in evthe model refinement step. Passed as argument "method" to `fit_poisson_nmf`.
`init.method`	The method used to initialize the factors and loadings. See `init_poisson_nmf` for details.
`control.init`	A list of parameters controlling the behaviour of the optimization and Topic SCORE method in the call to `init_poisson_nmf`. This is passed as argument "control" to `init_poisson_nmf`.

`control.main` A list of parameters controlling the behaviour of the optimization in the main model fitting step. This is passed as argument "control" to `fit_poisson_nmf`.

`control.refine` A list of parameters controlling the behaviour of the of the optimization algorithm in the model refinement step. This is passed as argument "control" to `fit_poisson_nmf`.

`verbose` When `verbose = "progressbar"` or `verbose = "detailed"`, information about the progress of the model fitting is printed to the console. See `fit_poisson_nmf` for more information.

Details

With the default settings, the model fitting is accomplished in four steps: (1) initialize the Poisson NMF model fit (`init_poisson_nmf`); (2) perform the main model fitting step by running 100 EM updates using `fit_poisson_nmf`; (3) refine the fit by running 100 extrapolated SCD updates, again using `fit_poisson_nmf`; and (4) recover the multinomial topic model by calling `poisson2multinom`.

This two-stage fitting approach is based on our findings that the EM algorithm initially makes rapid progress toward a solution, but its convergence slows considerably as the iterates approach a solution. Close to a solution, we have found that other algorithms make much more rapid progress than EM; in particularly, we found that the extrapolated SCD updates usually performed best). For larger data sets, more updates in the main model fitting and refinement steps may be needed to obtain a good fit.

For larger data sets, more than 200 updates may be needed to obtain a good fit.

Value

A multinomial topic model fit; see `poisson2multinom` and `fit_poisson_nmf` for details. Note that outputted likelihoods and deviances in `progress` are evaluated with respect to the equivalent Poisson NMF model.

References

Dey, K. K., Hsiao, C. J. and Stephens, M. (2017). Visualizing the structure of RNA-seq expression data using grade of membership models. *PLoS Genetics* **13**, e1006599.

Blei, D. M., Ng, A. Y. and Jordan, M. I. (2003). Latent Dirichlet allocation. *Journal of Machine Learning Research* **3**, 993-1022.

Hofmann, T. (1999). Probabilistic latent semantic indexing. In *Proceedings of the 22nd International ACM SIGIR Conference*, 50-57. doi:10.1145/312624.312649

See Also

`init_poisson_nmf`, `fit_poisson_nmf`, `poisson2multinom`, `fit_multinom_model`

Examples

```
library(Matrix)
set.seed(1)
X <- simulate_count_data(80,100,k = 3,sparse = TRUE)$X
```

```
fit <- fit_topic_model(X,k = 3)
print(summary(fit))
```

`loadings_plot` *Loadings Plot*

Description

Generate one or more barcharts to visualize the relationship between the loadings or mixture proportions and a selected categorical variable (a factor).

Usage

```
loadings_plot(
  fit,
  x,
  k,
  ggplot_call = loadings_plot_ggplot_call,
  plot_grid_call = function(plots) do.call(plot_grid, plots)
)

loadings_plot_ggplot_call(dat, topic.label, font.size = 9)
```

Arguments

`fit`	An object of class "poisson_nmf_fit" or "multinom_topic_model_fit".
`x`	A categorical variable represented as a `factor`. It should have the same number of elements as the number of rows in `fit$L`.
`k`	The topic, or topics, selected by number or name. When not specified, all topics are plotted.
`ggplot_call`	The function used to create the plot. Replace `loadings_plot_ggplot_call` with your own function to customize the appearance of the plot.
`plot_grid_call`	When multiple topics are selected, this is the function used to arrange the plots into a grid using `plot_grid`. It should be a function accepting a single argument, `plots`, a list of `ggplot` objects.
`dat`	A data frame passed as input to `ggplot`, containing, at a minimum, columns "x" and "loading".
`topic.label`	The name or number of the topic being plotted. Only used to determine the plot title.
`font.size`	Font size used in plot.

Details

This is a lightweight interface primarily intended to expedite creation of boxplots for investigating relationships between topics and a categorical variables of interest without having to spend a great deal of time worrying about the plotting settings; most of the "heavy lifting" is done by 'ggplot2' (specifically, function `geom_boxplot` in the 'ggplot2' package). For more control over the plot's appearance, the plot can be customized by modifying the `ggplot_call` and `plot_grid_call` arguments.

Value

A `ggplot` object.

`loglik_poisson_nmf` *NMF and Topic Model Likelihoods and Deviances*

Description

Compute log-likelihoods and deviances for assessing fit of a topic model or a non-negative matrix factorization (NMF).

Usage

```
loglik_poisson_nmf(X, fit, e = 1e-08)

loglik_multinom_topic_model(X, fit, e = 1e-08)

deviance_poisson_nmf(X, fit, e = 1e-08)

cost(X, A, B, e = 1e-08, family = c("poisson", "multinom"), version)
```

Arguments

`X`	The n x m matrix of counts or pseudocounts. It can be a sparse matrix (class `"dgCMatrix"`) or dense matrix (class `"matrix"`).
`fit`	A Poisson NMF or multinomial topic model fit, such as an output from `fit_poisson_nmf` or `fit_topic_model`.
`e`	A small, non-negative number added to the terms inside the logarithms to avoid computing logarithms of zero. This prevents numerical problems at the cost of introducing a very small inaccuracy in the computation.
`A`	The n x k matrix of loadings. It should be a dense matrix.
`B`	The k x m matrix of factors. It should be a dense matrix.
`family`	If `model = "poisson"`, the loss function values corresponding to the Poisson non-negative matrix factorization are computed; if `model = "multinom"`, the multinomial topic model loss function values are returned.

version — When version == "R", the computations are performed entirely in R; when version == "Rcpp", an Rcpp implementation is used. The R version is typically faster when X is a dense matrix, whereas the Rcpp version is faster and more memory-efficient when X is a large, sparse matrix. When not specified, the most suitable version is called depending on whether X is dense or sparse.

Details

Function cost computes loss functions proportional to the negative log-likelihoods, and is mainly for internal use to quickly compute log-likelihoods and deviances; it should not be used directly unless you know what you are doing. In particular, little argument checking is performed by cost.

Value

A numeric vector with one entry per row of X.

Examples

```
# Generate a small counts matrix.
set.seed(1)
out <- simulate_count_data(10,20,3)
X   <- out$X
fit <- out[c("F","L")]
class(fit) <- c("poisson_nmf_fit","list")

# Compute the Poisson log-likelihoods and deviances.
data.frame(loglik   = loglik_poisson_nmf(X,fit),
           deviance = deviance_poisson_nmf(X,fit))

# Compute multinomial log-likelihoods.
loglik_multinom_topic_model(X,fit)
```

---

merge_topics *Combine Topics in Multinomial Topic Model*

Description

Combine two or more topics in a multinomial topic model fit.

Usage

```
merge_topics(fit, k)
```

Arguments

fit — A multinomial topic model fit.

k — The names or numbers of the topics to be combined. Two or more topics should be chosen.

Details

Mixture proportions are combined by summation, and factors are combined by averaging.

Value

A multinomial topic model fit.

---

multinom2poisson *Recover Poisson NMF Fit from Multinomial Topic Model Fit*

Description

This function recovers parameter estimates of the Poisson non-negative matrix factorization (NMF) given parameter estimates for a multinomial topic model.

Usage

```
multinom2poisson(fit, X)
```

Arguments

`fit`	An object of class "multinom_topic_model_fit", such as an output from `poisson2multinom`. If a Poisson NMF fit is provided (that is, an object of class "poisson_nmf_fit"), the fit object is immediately returned "as is".
`X`	Optional n x m matrix of counts, or pseudocounts. It can be a sparse matrix (class `"dgCMatrix"`) or dense matrix (class `"matrix"`). This only needs to be provided if the document sizes `fit$s` are not available.

Value

The return value is the list `fit`, in which matrices `fit$F` and `fit$L` specify the factors and loadings in the Poisson non-negative matrix factorization; specifically, the counts matrix is modeled by the low-rank matrix product `tcrossprod(fit$L,fit$F)`.

---

pbmc_facs *Mixture of 10 FACS-purified PBMC Single-Cell RNA-seq data*

Description

These data are a selection of the reference transcriptome profiles generated via single-cell RNA sequencing (RNA-seq) of 10 bead-enriched subpopulations of PBMCs (Donor A), described in Zheng *et al* (2017). The data are unique molecular identifier (UMI) counts for 16,791 genes in 3,774 cells. (Genes with no expression in any of the cells were removed.) Since the majority of the UMI counts are zero, they are efficiently stored as a 3,774 x 16,791 sparse matrix. These data are used in the vignette illustrating how 'fastTopics' can be used to analyze to single-cell RNA-seq data. Data for a separate set of 1,000 cells is provided as a "test set" to evaluate out-of-sample predictions.

Format

pbmc_facs is a list with the following elements:

counts 3,774 x 16,791 sparse matrix of UMI counts, with rows corresponding to samples (cells) and columns corresponding to genes. It is an object of class "dgCMatrix").

counts_test UMI counts for an additional test set of 100 cells.

samples Data frame containing information about the samples, including cell barcode and source FACS population ("celltype" and "facs_subpop").

samples_test Sample information for the additional test set of 100 cells.

genes Data frame containing information and the genes, including gene symbol and Ensembl identifier.

fit Poisson non-negative matrix factorization (NMF) fitted to the UMI count data counts, with rank k = 6. See the vignette how the Poisson NMF model fitting was performed.

de Result of calling de_analysis(fit,counts,pseudocount = 0.1, control = list(ns = 1e4,nc = 4)) after first setting the seed to 1, set.seed(1).

Source

https://www.10xgenomics.com/resources/datasets

References

G. X. Y. Zheng *et al* (2017). Massively parallel digital transcriptional profiling of single cells. *Nature Communications* **8**, 14049. doi: 10.1038/ncomms14049

Examples

```
library(Matrix)
data(pbmc_facs)
cat(sprintf("Number of cells: %d\n",nrow(pbmc_facs$counts)))
cat(sprintf("Number of genes: %d\n",ncol(pbmc_facs$counts)))
cat(sprintf("Proportion of counts that are non-zero: %0.1f%%.\n",
            100*mean(pbmc_facs$counts > 0)))
```

---

pca_from_topics *Low-dimensional Embeddings from Poisson NMF or Multinomial Topic Model*

Description

Lightweight interface for rapidly producing low-dimensional embeddings from matrix factorizations or multinomial topic models. The defaults used are more suitable for producing embeddings from matrix factorizations or topic models.

Usage

```
pca_from_topics(fit, dims = 2, center = TRUE, scale. = FALSE, ...)

tsne_from_topics(
  fit,
  dims = 2,
  pca = FALSE,
  normalize = FALSE,
  perplexity = 100,
  theta = 0.1,
  max_iter = 1000,
  eta = 200,
  check_duplicates = FALSE,
  verbose = TRUE,
  ...
)

umap_from_topics(
  fit,
  dims = 2,
  n_neighbors = 30,
  metric = "euclidean",
  scale = "none",
  pca = NULL,
  verbose = TRUE,
  ...
)
```

Arguments

fit	An object of class "poisson_nmf_fit" or "multinom_topic_model_fit".
dims	The number of dimensions in the embedding. In tsne_from_topics, this is passed as argument "dims" to Rtsne. In umap_from_topics, this is passed as argument "n_components" to umap.
center	A logical value indicating whether columns of fit$L should be zero-centered before performing PCA; passed as argument "center" to prcomp.
scale.	A logical value indicating whether columns of fit$L should be scaled to have unit variance prior to performing PCA; passed as argument "scale." to prcomp.
...	Additional arguments passed to prcomp, Rtsne or umap.
pca	Whether to perform a PCA processing step in t-SNE or UMAP; passed as argument "pca" to Rtsne or umap.
normalize	Whether to normalize the data prior to running t-SNE; passed as argument "normalize" to Rtsne.
perplexity	t-SNE perplexity parameter, passed as argument "perplexity" to Rtsne. The perplexity is automatically revised if it is too large; see Rtsne for more information.
theta	t-SNE speed/accuracy trade-off parameter; passed as argument "theta" to Rtsne.

max_iter	Maximum number of t-SNE iterations; passed as argument "max_iter" to Rtsne.
eta	t-SNE learning rate parameter; passed as argument "eta" to Rtsne.
check_duplicates	When check_duplicates = TRUE, checks whether there are duplicate rows in fit$L; passed as argument "check_duplicates" to Rtsne.
verbose	If verbose = TRUE, progress updates are printed; passed as argument "verbose" to Rtsne or umap.
n_neighbors	Number of nearest neighbours in manifold approximation; passed as argument "n_neighbors" to umap.
metric	Distance matrix used to find nearest neighbors; passed as argument "metric" to umap.
scale	Scaling to apply to fit$L; passed as argument "scale" to umap.

Details

Note that since tsne_from_topics and umap_from_topics use nonlinear transformations of the data, distances between points are generally less interpretable than a linear transformation obtained by, say, PCA.

Value

An n x d matrix containing the embedding, where n is the number of rows of fit$L, and d = dims.

References

Kobak, D. and Berens, P. (2019). The art of using t-SNE for single-cell transcriptomics. *Nature Communications* **10**, 5416. doi: 10.1038/s4146701913056x

See Also

pca_plot, tsne_plot, umap_plot, prcomp, Rtsne, umap

Examples

```
library(ggplot2)
library(cowplot)
set.seed(1)
data(pbmc_facs)

# Get the Poisson NMF and multinomial topic model fit to the PBMC data.
fit1 <- multinom2poisson(pbmc_facs$fit)
fit2 <- pbmc_facs$fit
fit2 <- poisson2multinom(fit1)

# Compute the first two PCs of the loadings matrix (for the topic
# model, fit2, the loadings are the topic proportions).
Y1 <- pca_from_topics(fit1)
Y2 <- pca_from_topics(fit2)
subpop <- pbmc_facs$samples$subpop
```

```
quickplot(Y1[,1],Y1[,2],color = subpop) + theme_cowplot()
quickplot(Y2[,1],Y2[,2],color = subpop) + theme_cowplot()

# Compute a 2-d embedding of the loadings using t-SNE.

Y1 <- tsne_from_topics(fit1)
Y2 <- tsne_from_topics(fit2)
quickplot(Y1[,1],Y1[,2],color = subpop) + theme_cowplot()
quickplot(Y2[,1],Y2[,2],color = subpop) + theme_cowplot()


# Compute a 2-d embedding of the loadings using UMAP.

Y1 <- umap_from_topics(fit1)
Y2 <- umap_from_topics(fit2)
quickplot(Y1[,1],Y1[,2],color = subpop) + theme_cowplot()
quickplot(Y2[,1],Y2[,2],color = subpop) + theme_cowplot()
```

`plot_loglik_vs_rank` *Plot Log-Likelihood Versus Rank*

Description

Create a plot showing the improvement in the log-likelihood as the rank of the matrix factorization or the number of topics ("k") increases.

Usage

```
plot_loglik_vs_rank(fits, ggplot_call = loglik_vs_rank_ggplot_call)

loglik_vs_rank_ggplot_call(dat, font.size = 9)
```

Arguments

`fits`	A list with 2 more list elements, in which each list element is an object of class `"poisson_nmf_fit"` or `"multinom_topic_model_fit"`. If two or more fits share the same rank, or number of topics, the largest log-likelihood is plotted.
`ggplot_call`	The function used to create the plot. Replace `loglik_vs_rank_ggplot_call` with your own function to customize the appearance of the plot.
`dat`	A data frame passed as input to `ggplot`, containing, at a minimum, columns "x" and "y".
`font.size`	Font size used in plot.

Value

A `ggplot` object.

plot_progress *Plot Progress of Model Fitting Over Time*

Description

Create a plot showing improvement in one or more Poisson NMF or multinomial topic model fits over time.

Usage

```
plot_progress(
  fits,
  x = c("timing", "iter"),
  y = c("loglik", "dev", "res"),
  add.point.every = 20,
  colors = c("#E69F00", "#56B4E9", "#009E73", "#F0E442", "#0072B2", "#D55E00",
    "#CC79A7"),
  linetypes = "solid",
  linesizes = 0.5,
  shapes = 19,
  fills = "white",
  e = 0.01,
  theme = function() theme_cowplot(12)
)
```

Arguments

fits	An object of class "poisson_nmf_fit" or "multinom_topic_model_fit", or a non-empty, named list in which each all list elements are objects of class "poisson_nmf_fit" or all objects of class "multinom_topic_model_fit".
x	Choose "timing" to plot improvement in the solution over time, or choose "iter" to plot improvement in the solution per iteration.
y	Column of the "progress" data frame used to assess progress of the Poisson NMF optimization method(s). Should be one of "loglik" (Poisson NMF or multinomial topic model log-likelihood), "dev" (deviance) or "res" (maximum residual of KKT conditions). The deviance is only valid for Poisson NMF model fits.
add.point.every	A positive integer giving the iteration interval for drawing points on the progress curves. Set to Inf to prevent points from being drawn on the plot.
colors	Colours used to draw progress curves; passed as the values input to scale_color_manual. If fewer colours than "fits" are given, the colours are recycled.
linetypes	Line types used to draw progress curves; passed as the values input to scale_linetype_manual. If fewer line types than "fits" are given, the line types are recycled.
linesizes	Line sizes used to draw progress curves; passed as the values input to scale_size_manual. If fewer line sizes than "fits" are given, the line sizes are recycled.

shapes	Shapes used to draw points at the selected iterations; passed as the values input to scale_shape_manual. If fewer shapes than "fits" are given, the shapes are recycled.
fills	Fill colours used to draw points at the selected iterations; passed as the values input to scale_fill_manual. If fewer fill colours than "fits" are given, the fill colours are recycled.
e	A small, positive number added to the vertical axis (for y = "loglik" and y = "dev" only) so that the logarithmic scale does not over-emphasize very small differences.
theme	The 'ggplot2' "theme".

Details

The horizontal axis shows the recorded runtime (in s), and the vertical axis shows some quantity measuring the quality of the fit: the log-likelihood, deviance or maximum residual of the Karush-Kuhn-Tucker (KKT) first-order optimality conditions. To better visualize log-likelihoods and deviances, log-likelihood and deviance differences are shown on the logarithmic scale. Differences are calculated with respect to the best value achieved over all the fits compared.

Note that only minimal argument checking is performed.

Value

A ggplot object.

See Also

fit_poisson_nmf

---

poisson2multinom *Recover Multinomial Topic Model Fit from Poisson NMF fit*

Description

This function recovers parameter estimates of the multinomial topic model given parameter estimates for a Poisson non-negative matrix factorization (NMF).

Usage

```
poisson2multinom(fit)
```

Arguments

fit	An object of class "poisson_nmf_fit", such as an output from fit_poisson_nmf. It does not make sense for a multinomial topic model to have less than two topics, so an error will be reported when k < 2, where k is the rank of the matrix factorization. If a multinomial topic model fit is provided (that is, an object of class "multinom_topic_model_fit"), the fit object is immediately returned "as is".

Value

The return value is the list `fit`, in which `fit$F` and `fit$L` are the parameters of the multinomial topic model; specifically, `fit$L[i,]` gives the topic probabilities for sample or document i, and `fit$F[,k]` gives the term probabilities for topic k. An additional vector `fit$s` of length n is returned giving the "size factors".

`predict.poisson_nmf_fit`

Predict Methods for Poisson NMF and Multinomial Topic Model

Description

Predict loadings based on previously fit Poisson NMF, or predict topic proportions based on previously fit multinomial topic model. This can be thought of as projecting data points onto a previously estimated set of factors `fit$F`.

Usage

```
## S3 method for class 'poisson_nmf_fit'
predict(object, newdata, numiter = 20, ...)

## S3 method for class 'multinom_topic_model_fit'
predict(object, newdata, numiter = 20, ...)
```

Arguments

`object`	An object of class “poisson_nmf_fit” or “multinom_topic_model_fit”.
`newdata`	An optional counts matrix. If omitted, the loadings estimated in the original data are returned.
`numiter`	The number of updates to perform.
`...`	Additional arguments passed to `fit_poisson_nmf`.

Value

A loadings matrix with one row for each data point and one column for each topic or factor. For `predict.multinom_topic_model_fit`, the output can also be interpreted as a matrix of estimated topic proportions, in which `L[i,j]` is the proportional contribution of topic j to data point i.

See Also

`fit_poisson_nmf`

Examples

```
# Simulate a 175 x 1,200 counts matrix.
set.seed(1)
dat <- simulate_count_data(175,1200,k = 3)

# Split the data into training and test sets.
train <- dat$X[1:100,]
test <- dat$X[101:175,]

# Fit a Poisson non-negative matrix factorization using the
# training data.
fit <- init_poisson_nmf(train,F = dat$F,init.method = "random")
fit <- fit_poisson_nmf(train,fit0 = fit)

# Compare the estimated loadings in the training data against the
# loadings used to simulate these data.
Ltrain <- predict(fit)
plot(dat$L[1:100,],Ltrain,pch = 20,col = "darkblue")
abline(a = 0,b = 1,col = "magenta",lty = "dotted",
       xlab = "true",ylab = "estimated")

# Next, predict loadings in unseen (test) data points, and compare
# these predictions against the loadings that were used to simulate
# the test data.
Ltest <- predict(fit,test)
plot(dat$L[101:175,],Ltest,pch = 20,col = "darkblue",
     xlab = "true",ylab = "estimated")
abline(a = 0,b = 1,col = "magenta",lty = "dotted")

# Simulate a 175 x 1,200 counts matrix.
set.seed(1)
dat <- simulate_multinom_gene_data(175,1200,k = 3)

# Split the data into training and test sets.
train <- dat$X[1:100,]
test <- dat$X[101:175,]

# Fit a topic model using the training data.
fit <- init_poisson_nmf(train,F = dat$F,init.method = "random")
fit <- fit_poisson_nmf(train,fit0 = fit)
fit <- poisson2multinom(fit)

# Compare the estimated topic proportions in the training data against
# the topic proportions used to simulate these data.
Ltrain <- predict(fit)
plot(dat$L[1:100,],Ltrain,pch = 20,col = "darkblue")
abline(a = 0,b = 1,col = "magenta",lty = "dotted",
       xlab = "true",ylab = "estimated")

# Next, predict loadings in unseen (test) data points, and compare
# these predictions against the loadings that were used to simulate
```

```
# the test data.
Ltest <- predict(fit,test)
plot(dat$L[101:175,],Ltest,pch = 20,col = "darkblue",
     xlab = "true",ylab = "estimated")
abline(a = 0,b = 1,col = "magenta",lty = "dotted")
```

`run_homer` *Perform HOMER Motif Enrichment Analysis using DE Genomic Positions*

Description

Run HOMER motif finding algorithm (`findMotifsGenome.pl`) to identify motifs enriched for differentially expressed (DE) genomic positions. See http://homer.ucsd.edu for more information.

Usage

```
run_homer(
  de,
  k,
  positions,
  genome = "hg19",
  subset = function(postmean, lpval, lfsr, rank, quantile) lfsr < 0.05,
  homer.exec = "findMotifsGenome.pl",
  out.dir = tempdir(),
  homer.options = "-len 8,10,12 -size 200 -mis 2 -S 25 -p 1 -h",
  verbose = TRUE
)
```

Arguments

de	An object of class "topic_model_de_analysis", usually the result of running de_analysis.
k	Use the DE analysis results for this topic.
positions	A table of genomic positions corresponding to rows of the de_analysis results. Specifically, it should a data frame with four columns: "chr", chromosome name or number; "start", start position of genomic feature; "end", end position of genomic feature; and "name", the name of the genomic feature. If not specified, the genomic positions will be extracted from the row names of de$postmean, in which the row names are expected to be of the form chr_start_end. The genomic positions will be written to a BED file (see https://genome.ucsc.edu/FAQ/FAQformat.html for more information about BED files).
genome	The genome parameter passed to findMotifsGenome.pl.
subset	Describe input argument "subset" here.

homer.exec	The name or file path of the HOMER findMotifsGenome.pl excutable.
out.dir	The positions BED file and HOMER results are written to this directory.
homer.options	Character string used to override default findMotifsGenome.pl options.
verbose	When verbose = TRUE, progress information is printed to the console.

Value

A data frame containing the motif enrichment results. It is created from the knownResults.txt HOMER output.

References

Heinz, S., Benner, C., Spann, N., Bertolino, E., Lin, Y. C., Laslo, P., Cheng, J. X., Murre, C., Singh, H. and Glass, C. K. (2010). Simple combinations of lineage-determining transcription factors prime cis-regulatory elements required for macrophage and B cell identities. *Molecular Cell* **38**, 576-589.

---

select.poisson_nmf_fit

Extract or Re-order Data Rows in Poisson NMF or Multinomial Topic Model Fit

---

Description

This function can be used to extract estimates for a subset of the count data, or to re-order the rows of the loadings matrix.

Usage

```
## S3 method for class 'poisson_nmf_fit'
select(.data, loadings, ...)

## S3 method for class 'multinom_topic_model_fit'
select(.data, loadings, ...)

select_loadings(.data, loadings, ...)
```

Arguments

.data	Poisson NMF or Multinomial Topic Model fit; that is, an object of class "poisson_nmf_fit" or "multinom_topic_model_fit", such as an output from fit_poisson_nmf or fit_topic_model.
loadings	Indices (names or numbers) giving data rows to keep. If not specified, all rows are kept.
...	Other arguments passed to the generic select function.

Value

A Poisson NMF or multinomial topic model fit containing the selected data rows only.

simulate_count_data *Simulate Count Data from Poisson NMF Model*

Description

Simulate a counts matrix `X` such that `X[i,j]` is Poisson with rate (mean) `Y[i,j]`, where `Y = tcrossprod(L,F)`, `L` is an n x k loadings ("activations") matrix, and `F` is an m x k factors ("basis vectors") matrix. The entries of matrix `L` are drawn uniformly at random between zero and `lmax`, and the entries of matrix `F` are drawn uniformly at random between 0 and `fmax`.

Usage

```
simulate_count_data(n, m, k, fmax = 1, lmax = 1, sparse = FALSE)
```

Arguments

`n`	Number of rows in simulated count matrix. The number of rows should be at least 2.
`m`	Number of columns in simulated count matrix. The number of columns should be at least 2.
`k`	Number of factors, or "topics", used to determine Poisson rates. The number of topics should be 1 or more.
`fmax`	Factors are drawn uniformly at random between zero and `fmax`.
`lmax`	Loadings are drawn uniformly at random between zero and `lmax`.
`sparse`	If `sparse = TRUE`, convert the counts matrix to a sparse matrix in compressed, column-oriented format; see `sparseMatrix`.

Details

Note that only minimal argument checking is performed. This function is mainly used to simulate small data sets for the examples and package tests.

Value

The return value is a list containing the counts matrix `X` and the factorization, `F` and `L`, used to generate the counts.

`simulate_poisson_gene_data` *Simulate Gene Expression Data from Poisson NMF or Multinomial Topic Model*

Description

Simulate count data from a Poisson NMF model or multinomial topic model, in which topics represent "gene expression programs", and gene expression programs are characterized by different rates of expression. The way in which the counts are simulated is modeled after gene expression studies in which expression is measured by single-cell RNA sequencing ("RNA-seq") techniques: each row of the counts matrix corresponds a gene expression profile, each column corresponds to a gene, and each matrix element is a "read count", or "UMI count", measuring expression level. Factors are simulated so as to capture realistic changes in gene expression across different cell types. See "Details" for the procedure used to simulate factors, loadings and counts.

Usage

```
simulate_poisson_gene_data(n, m, k, s, p = 1, sparse = FALSE)

simulate_multinom_gene_data(n, m, k, sparse = FALSE)
```

Arguments

`n`	Number of rows in the simulated count matrix. Should be at least 2.
`m`	Number of columns in the simulated count matrix. Should be at least 2.
`k`	Number of factors, or "topics", used to generate the data. Should be 2 or more.
`s`	Vector of "size factors"; each row of the loadings matrix `L` is scaled by the entries of `s` before generating the counts. This should be a vector of length n containing only positive values.
`p`	Probability that `F[i,j]` is equal to the mean rate. Smaller values of `p` will result in more factors that are the same across topics.
`sparse`	If `sparse = TRUE`, convert the counts matrix to a sparse matrix in compressed, column-oriented format; see `sparseMatrix`.

Details

Here we describe the process for generating the n x k loadings matrix `L` and the m x k factors matrix `F`.

Each row of the `L` matrix is generated in the following manner: (1) the number of nonzero mixture proportions is $1 \leq n \leq k$, with probability proportional to 2^{-n}; (2) the indices of the nonzero mixture proportions are sampled uniformly at random; and (3) the nonzero mixture proportions are sampled from the Dirichlet distribution with $\alpha = 1$ (so that all topics are equally likely).

Each row of the factors matrix are generated according to the following procedure: (1) generate $u = |r| - 5$, where r $N(0, 2)$; (2) for each topic k, generate the Poisson rates as $exp(max(t, -5))$,

where t $0.95 * N(u, s/10) + 0.05 * N(u, s)$, and $s = exp(-u/8)$. Factors can be interpreted as Poisson rates or multinomial probabilities, so that individual counts can be viewed as being generated from a weighted mixture of "topics" with different rates or probabilities.

Once the loadings and factors have been generated, the counts are simulated from either the Poisson NMF or multinomial topic model: for the former, `X[i,j]` is Poisson with rate `Y[i,j]`, where `Y = tcrossprod(L,F)`; for the latter, `X[i,]` is multinomial with size `s[i]` and with class probabilities `P[i,]`, where `P = tcrossprod(L,F)`. For the multinomial model only, the sizes `s` are randomly generated as `s = 10^rnorm(n,3,0.2)`.

Note that only minimal argument checking is performed; the function is mainly used to test implementation of the topic-model-based differential count analysis.

Value

`simulate_poisson_gene_data` returns a list containing the counts matrix `X`, and the size factors `s` and factorization, `F`, `L`, used to generate the counts. `simulate_multinom_gene_data` returns a list containing the counts matrix `X`, and the mixture proportions `L` and factors (gene probabilities, or relative gene expression levels) `F` used to generate the counts.

---

simulate_toy_gene_data

Simulate Toy Gene Expression Data

Description

Simulate gene expression data (UMI counts) under a toy expression model. Samples (expression profiles) are drawn from a multinomial topic model in which topics are "gene programs".

Usage

```
simulate_toy_gene_data(n, m, k, s)
```

Arguments

`n`	The number of samples (gene expression profiles) to simulate.
`m`	The number of counts (genes) to simulate.
`k`	The number of topics ("gene programs") used to simulate the data.
`s`	A scalar specifying the total expression of each sample; it specifies the "size" parameter in the calls to `rmultinom`.

Details

The mixture proportions are generated as follows. With probability 0.9, one proportion is one, or close to one, and the remaining are zero, or close to zero; that is, the counts are primarily generated from a single gene program. Otherwise (wtth probability 0.1), the mixture proportions are roughly equal.

Gene frequencies are drawn uniformly at random from [0,1].

Value

The return value is a list containing the counts matrix `X`, and the gene frequencies `F` and mixture proportions `L` used to generate the counts.

`structure_plot` *Structure Plot*

Description

Create a "Structure plot" from a multinomial topic model fit. The Structure plot represents the estimated topic proportions of each sample in a stacked bar chart, with bars of different colors representing different topics. Consequently, samples that have similar topic proportions have similar amounts of each color.

Usage

```
structure_plot(
  fit,
  topics,
  grouping,
  loadings_order = "embed",
  n = 2000,
  colors,
  gap = 1,
  embed_method = structure_plot_default_embed_method,
  ggplot_call = structure_plot_ggplot_call,
  ...
)

structure_plot_default_embed_method(fit, ...)

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

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

structure_plot_ggplot_call(dat, colors, ticks = NULL, font.size = 9)
```

Arguments

`fit`	An object of class "poisson_nmf_fit" or "multinom_topic_model_fit". If a Poisson NMF fit is provided as input, the corresponding multinomial topic model fit is automatically recovered using `poisson2multinom`.

topics	Top-to-bottom ordering of the topics in the Structure plot; topics[1] is shown on the top, topics[2] is shown next, and so on. If the ordering of the topics is not specified, the topics are automatically ordered so that the topics with the greatest total "mass" are at shown at the bottom of the plot. The topics may be specified by number or by name.
grouping	Optional categorical variable (a factor) with one entry for each row of the loadings matrix fit$L defining a grouping of the samples (rows). The samples (rows) are arranged along the horizontal axis according to this grouping, then within each group according to loadings_order. If grouping is not a factor, an attempt is made to convert it to a factor using as.factor. Note that if loadings_order is specified manually, grouping should be the groups for the rows of fit$L *before* reordering.
loadings_order	Ordering of the rows of the loadings matrix fit$L along the horizontal axis the Structure plot (after they have been grouped). If loadings_order = "embed", the ordering is generated automatically from a 1-d embedding, separately for each group. The rows may be specified by number or by name. Note that loadings_order may include all the rows of fit$L, or a subset.
n	The maximum number of samples (rows of the loadings matrix fit$L) to include in the plot. Typically there is little to no benefit in including large number of samples in the Structure plot due to screen resolution limits. Ignored if loadings_order is provided.
colors	Colors used to draw topics in Structure plot.
gap	The horizontal spacing between groups. Ignored if grouping is not provided.
embed_method	The function used to compute an 1-d embedding from a loadings matrix fit$L; only used if loadings_order = "embed". The function must accept the multinomial topic model fit as its first input ("fit") and additional arguments may be passed (...). The output should be a named numeric vector with one entry per row of fit$L, and the names of the entries should be the same as the row names of fit$L.
ggplot_call	The function used to create the plot. Replace structure_plot_ggplot_call with your own function to customize the appearance of the plot.
...	Additional arguments passed to structure_plot (for the plot method) or embed_method (for function structure_plot).
x	An object of class "poisson_nmf_fit" or "multinom_topic_model_fit". If a Poisson NMF fit is provided as input, the corresponding multinomial topic model fit is automatically recovered using poisson2multinom.
dat	A data frame passed as input to ggplot, containing, at a minimum, columns "sample", "topic" and "prop": the "sample" column contains the positions of the samples (rows of the L matrix) along the horizontal axis; the "topic" column is a topic (a column of L); and the "prop" column is the topic proportion for the respective sample.
ticks	The placement of the group labels along the horizontal axis, and their names. For data that are not grouped, use ticks = NULL.
font.size	Font size used in plot.

Details

The name "Structure plot" comes from its widespread use in population genetics to visualize the results of the Structure software (Rosenberg *et al*, 2002).

For most uses of the Structure plot in population genetics, there is usually some grouping of the samples (e.g., assignment to pre-defined populations) that guides arrangement of the samples along the horizontal axis in the bar chart. In other applications, such as analysis of gene expression data, a pre-defined grouping may not always be available. Therefore, a "smart" arrangement of the samples is, by default, generated automatically by performing a 1-d embedding of the samples.

Value

A `ggplot` object.

References

Dey, K. K., Hsiao, C. J. and Stephens, M. (2017). Visualizing the structure of RNA-seq expression data using grade of membership models. *PLoS Genetics* **13**, e1006599.

Rosenberg, N. A., Pritchard, J. K., Weber, J. L., Cann, H. M., Kidd, K. K., Zhivotovsky, L. A. and Feldman, M. W. (2002). Genetic structure of human populations. *Science* **298**, 2381–2385.

Examples

```
set.seed(1)
data(pbmc_facs)

# Get the multinomial topic model fitted to the
# PBMC data.
fit <- pbmc_facs$fit

# Create a Structure plot without labels. The samples (rows of L) are
# automatically arranged along the x-axis using t-SNE to highlight the
# structure in the data.
p1 <- structure_plot(fit)

# Create a Structure plot with the FACS cell-type labels. Within each
# group (cell-type), the cells (rows of L) are automatically arranged
# using t-SNE.
subpop <- pbmc_facs$samples$subpop
p2 <- structure_plot(fit,grouping = subpop)

# Next, we apply some customizations to improve the plot: (1) use the
# "topics" argument to specify the order in which the topic
# proportions are stacked on top of each other; (2) use the "gap"
# argrument to increase the whitespace between the groups; (3) use "n"
# to decrease the number of rows of L included in the Structure plot;
# and (4) use "colors" to change the colors used to draw the topic
# proportions.
topic_colors <- c("skyblue","forestgreen","darkmagenta",
                  "dodgerblue","gold","darkorange")
```

```
p3 <- structure_plot(fit,grouping = pbmc_facs$samples$subpop,gap = 20,
                     n = 1500,topics = c(5,6,1,4,2,3),colors = topic_colors)

# In this example, we use UMAP instead of t-SNE to arrange the
# cells in the Structure plot. Note that this can be accomplished in
# a different way by overriding the default setting of
# "embed_method".
y <- drop(umap_from_topics(fit,dims = 1))
p4 <- structure_plot(fit,loadings_order = order(y),grouping = subpop,
                     gap = 40,colors = topic_colors)

# We can also use PCA to arrange the cells.
y <- drop(pca_from_topics(fit,dims = 1))
p5 <- structure_plot(fit,loadings_order = order(y),grouping = subpop,
                     gap = 40,colors = topic_colors)

# In this final example, we plot a random subset of 400 cells, and
# arrange the cells randomly along the horizontal axis of the
# Structure plot.
p6 <- structure_plot(fit,loadings_order = sample(3744,400),gap = 10,
                     grouping = subpop,colors = topic_colors)
```

`summary.poisson_nmf_fit` *Summarize Poisson NMF or Multinomial Topic Model Fit*

Description

`summary` method for the "poisson_nmf_fit" and "multinom_topic_model_fit" classes.

Usage

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

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

## S3 method for class 'summary.poisson_nmf_fit'
print(x, show.mixprops = FALSE, show.topic.reps = FALSE, ...)

## S3 method for class 'summary.multinom_topic_model_fit'
print(
  x,
  show.size.factors = FALSE,
  show.mixprops = FALSE,
  show.topic.reps = FALSE,
```

```
  ...
)
```

Arguments

object	An object of class "poisson_nmf_fit" or "multinom_topic_model_fit". The former is usually the result of calling fit_poisson_nmf; the latter is usually the result of calling fit_topic_model or poisson2multinom.
...	Additional arguments passed to the generic summary or print.summary method.
x	An object of class "summary.poisson_nmf_fit", usually a result of a call to summary.poisson_nmf_fit.
show.mixprops	If TRUE, print a summary of the mixture proportions.
show.topic.reps	If TRUE, print a summary of the topic representatives.
show.size.factors	If TRUE, print a summary of the size factors.

Value

The functions summary.poisson_nmf_fit and summary.multinom_topic_model_fit compute and return a list of statistics summarizing the model fit. The returned list includes some or all of the following elements:

n	The number of rows in the counts matrix, typically the number of samples.
m	The number of columns in the counts matrix, typically the number of observed counts per sample.
k	The rank of the Poisson NMF or the number of topics.
s	A vector of length n giving the "size factor" estimates; these estimates should be equal, or close to, the total counts in each row of the counts matrix.
numiter	The number of loadings and/or factor updates performed.
loglik	The Poisson NMF log-likelihood.
loglik.multinom	The multinomial topic model log-likelihood.
dev	The Poisson NMF deviance.
res	The maximum residual of the Karush-Kuhn-Tucker (KKT) first-order optimality conditions. This can be used to assess convergence of the updates to a (local) solution.
mixprops	Matrix giving a high-level summary of the mixture proportions, in which rows correspond to topics, and columns are ranges of mixture proportionss.
topic.reps	A matrix in which the ith row gives the mixture proportions for the sample "most representative" of topic i; by "most representative", we mean the row (or sample) with the highest proportion of counts drawn from the topic i.

volcano_plot *Volcano Plots for Visualizing Results of Differential Expression Analysis*

Description

Create a "volcano" plot to visualize the results of a differential count analysis using a topic model. Here, the volcano plot is a scatterplot in which the posterior mean log-fold change (LFC), estimated by running the methods implemented in de_analysis, is plotted against the estimated z-score. Variations on this volcano plot may also be created, for example by showing f0 (the null-model estimates) instead of the z-scores. Use volcano_plotly to create an interactive volcano plot.

Usage

```
volcano_plot(
  de,
  k,
  labels,
  y = c("z", "f0"),
  do.label = volcano_plot_do_label_default,
  ymin = 1e-06,
  ymax = Inf,
  max.overlaps = Inf,
  plot.title = paste("topic", k),
  ggplot_call = volcano_plot_ggplot_call
)

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

volcano_plotly(
  de,
  k,
  file,
  labels,
  y = c("z", "f0"),
  ymin = 1e-06,
  ymax = Inf,
  width = 500,
  height = 500,
  plot.title = paste("topic", k),
  plot_ly_call = volcano_plot_ly_call
)

volcano_plot_do_label_default(lfc, y)

volcano_plot_ggplot_call(dat, y, plot.title, max.overlaps = Inf, font.size = 9)
```

```
volcano_plot_ly_call(dat, y, plot.title, width, height)
```

Arguments

de	An object of class "topic_model_de_analysis", usually an output from de_analysis. It is better to run de_analysis with shrink.method = "ash" so that the points in the volcano plot can be coloured by their local false sign rate (lfsr).
k	The topic, selected by number or name.
labels	Character vector specifying how the points in the volcano plot are labeled. This should be a character vector with one entry per LFC estimate (row of de$postmean). When not specified, the row names of de$postmean are used. When available. labels are added to the plot using geom_text_repel.
y	A vector of the same length as lfc.
do.label	The function used to deetermine which LFC estimates to label. Replace volcano_plot_do_label_defau with your own function to customize the labeling of points in the volcano plot.
ymin	Y-axis values less than ymin are shown as ymin.
ymax	Y-axis values greater than ymax are shown as ymax. When y = "z", setting ymax to a finite value can improve the volcano plot when some z-scores are much larger (in magnitude) than others.
max.overlaps	Argument passed to geom_text_repel.
plot.title	The title of the plot.
ggplot_call	The function used to create the plot. Replace volcano_plot_ggplot_call with your own function to customize the appearance of the plot.
x	An object of class "topic_model_de_analysis", usually an output from de_analysis.
...	Additional arguments passed to volcano_plot.
file	Save the interactive volcano plot to this HTML file using saveWidget.
width	Width of the plot in pixels. Passed as argument "width" to plot_ly.
height	Height of the plot in pixels. Passed as argument "height" to plot_ly.
plot_ly_call	The function used to create the plot. Replace volcano_plot_ly_call with your own function to customize the appearance of the interactive plot.
lfc	A vector of log-fold change estimates.
dat	A data frame passed as input to ggplot, containing, at a minimum, columns "f0", "postmean", "y", "lfsr" and "label".
font.size	Font size used in plot.

Details

Interactive volcano plots can be created using the 'plotly' package. The "hover text" shows the label and detailed LFC statistics.

Value

A ggplot object or a plotly object.

See Also

de_analysis

Examples

```
# See help(de_analysis) for examples.
```

Index