Package ‘TCA’

November 16, 2019

Type Package

Title Tensor Composition Analysis

Version 1.1.0

Author Elior Rahmani [aut, cre]

Maintainer Elior Rahmani <elior.rahmani@gmail.com>

Description

Tensor Composition Analysis (TCA) allows the deconvolution of two-dimensional data (features by observations) coming from a mixture of sources into a three-dimensional matrix of signals (features by observations by sources). TCA further allows to test the features in the data for different statistical relations with an outcome of interest while modeling source-specific effects (TCA regression); particularly, it allows to look for statistical relations between source-specific signals and an outcome. For example, TCA can deconvolve bulk tissue-level DNA methylation data (methylation sites by individuals) into a tensor of cell-type-specific methylation levels for each individual (methylation sites by individuals by cell types) and it allows to detect cell-type-specific relations (associations) with an outcome of interest. For more details see Rahmani et al. (2018) <DOI:10.1101/437368>.

License GPL-3

Encoding UTF-8

LazyData true

Imports config, data.table, futile.logger, gmodels, Matrix, matrixcalc, matrixStats, nloptr, parallel, pbapply, pracma, rsvd, stats, quadprog, glmnet

RoxygenNote 6.1.1

Depends R (>= 3.4.0)

Suggests testthat, knitr, rmarkdown

URL https://www.nature.com/articles/s41467-019-11052-9

BugReports https://github.com/cozygene/TCA/issues

VignetteBuilder knitr

NeedsCompilation no

Repository CRAN

Date/Publication 2019-11-16 17:10:02 UTC

R topics documented:

refactor 2
tca 4
tcareg 8
tcasub 11
tensor 13
test_data 14

Index **16**

`refactor` *Sparse principal component analysis using ReFACTor*

Description

Performs unsupervised feature selection followed by principal component analysis (PCA) under a row-sparse model using the ReFACTor algorithm. For example, in the context of tissue-level bulk DNA methylation data coming from a mixture of cell types (i.e. the input is methylation sites by individuals), `refactor` allows to capture the variation in cell-type composition, which was shown to be a dominant sparse signal in methylation data.

Usage

```
refactor(X, k, sparsity = 500, C = NULL, C.remove = FALSE,
  sd_threshold = 0.02, num_comp = NULL, rand_svd = FALSE,
  log_file = "TCA.log", debug = FALSE, verbose = TRUE)
```

Arguments

`X`	An `m` by `n` matrix of measurements of `m` features for `n` observations. Each column in `X` is assumed to be a mixture of `k` different sources. Note that `X` must include row names and column names and that NA values are currently not supported. `X` should not include features that are constant across all observations.
`k`	A numeric value indicating the dimension of the signal in the data (i.e. the number of sources).
`sparsity`	A numeric value indicating the sparsity of the signal in the data (the number of signal rows).
`C`	An `n` by `p` design matrix of covariates that will be accounted for in the feature selection step. Note that `C` must include row names and column names and that NA values are currently not supported; ; set `C` to be `NULL` if there are no such covariates.
`C.remove`	A logical value indicating whether the covariates in X should be accounted for not only in the feature selection step, but also in the final calculation of the principal components (i.e. if `C.remove == TRUE` then the selected features will be adjusted for the covariates in `C` prior to calculating principal components). Note that setting `C.remove` to be `TRUE` is desired when ReFACTor is intended to

be used for correction in downstream analysis, whereas setting `C.remove` to be `FALSE` is desired when ReFACTor is merely used for capturing the sparse signals in the data (i.e. regardless of correction).

`sd_threshold` A numeric value indicating a standard deviation threshold to be used for excluding low-variance features in `X` (i.e. features with standard deviation lower than `sd_threshold` will be excluded). Set `sd_threshold` to be `NULL` for turning off this filter. Note that removing features with very low variability tends to improve speed and performance.

`num_comp` A numeric value indicating the number of ReFACTor components to return.

`rand_svd` A logical value indicating whether to use random svd for estimating the low-rank structure of the data in the first step of the algorithm; random svd can result in a substantial speedup for large data.

`log_file` A path to an output log file. Note that if the file `log_file` already exists then logs will be appended to the end of the file. Set `log_file` to `NULL` to prevent output from being saved into a file; note that if `verbose == TRUE` then no output file will be generated regardless of the value of `log_file`.

`debug` A logical value indicating whether to set the logger to a more detailed debug level; please set `debug` to `TRUE` before reporting issues.

`verbose` A logical value indicating whether to print logs.

Details

ReFACTor is a two-step algorithm for sparse principal component analysis (PCA) under a row-sparse model. The algorithm performs an unsupervised feature selection by ranking the features based on their correlation with their values under a low-rank representation of the data, followed by a calculation of principal components using the top ranking features (ReFACTor components).

Note that ReFACTor is tuned towards capturing sparse signals of the dominant sources of variation in the data. Therefore, in the presence of other potentially dominant factors in the data (i.e. beyond the variation of interest), these factors should be accounted for by including them as covariates (see argument `C`). In cases where the ReFACTor components are designated to be used as covariates in a downstream analysis alongside the covariates in `C` (e.g., in a standard regression analysis or in a TCA regression), it is advised to set the argument `C.remove` to be `TRUE`. This will adjust the selected features for the information in `C` prior to the calculation of the ReFACTor components, which will therefore capture only signals that is not present in `C` (and as a result may benefit the downstream analysis by potentially capturing more signals beyond the information in `C`).

Value

A list with the estimated components of the ReFACTor model.

`scores` An `n` by `num_comp` matrix of the ReFACTor components (the projection scores).

`coeffs` A `sparsity` by `num_comp` matrix of the coefficients of the ReFACTor components (the projection loadings).

`ranked_list` A vector with the features in the data, ranked by their scores in the feature selection step of the algorithm; the top scoring features (set according to the argument `sparsity`) are used for calculating the ReFACTor components. Note that features that were excluded according to `sd_threshold` will not appear in this `ranked_list`.

Note

For very large input matrices it is advised to use random svd for speeding up the feature selection step (see argument `rand_svd`).

References

Rahmani E, Zaitlen N, Baran Y, Eng C, Hu D, Galanter J, Oh S, Burchard EG, Eskin E, Zou J, Halperin E. Sparse PCA corrects for cell type heterogeneity in epigenome-wide association studies. Nature Methods 2016.

Rahmani E, Zaitlen N, Baran Y, Eng C, Hu D, Galanter J, Oh S, Burchard EG, Eskin E, Zou J, Halperin E. Correcting for cell-type heterogeneity in DNA methylation: a comprehensive evaluation. Nature Methods 2017.

Examples

```
data <- test_data(100, 200, 3, 0, 0, 0.01)
ref <- refactor(data$X, k = 3, sparsity = 50)
```

`tca` *Fitting the TCA model*

Description

Fits the TCA model for an input matrix of observations coming from a mixture of `k` sources, under the assumption that each observation is a mixture of unique source-specific values (in each feature in the data). For example, in the context of tissue-level bulk DNA methylation data coming from a mixture of cell types (i.e. the input is methylation sites by individuals), `tca` allows to model the methylation of each individual as a mixture of cell-type-specific methylation levels that are unique to the individual.

Usage

```
tca(X, W, C1 = NULL, C1.map = NULL, C2 = NULL, refit_W = FALSE,
  refit_W.features = NULL, refit_W.sparsity = 500,
  refit_W.sd_threshold = 0.02, tau = NULL, parallel = FALSE,
  num_cores = NULL, max_iters = 10, log_file = "TCA.log",
  debug = FALSE, verbose = TRUE)
```

Arguments

`X`	An `m` by `n` matrix of measurements of `m` features for `n` observations. Each column in `X` is assumed to be a mixture of `k` different sources. Note that `X` must include row names and column names and that NA values are currently not supported. `X` should not include features that are constant across all observations.

`W` An `n` by `k` matrix of weights - the weights of `k` sources for each of the `n` mixtures (observations). All the weights must be positive and each row, corresponding to the weights of a single observation, must sum up to 1. Note that `W` must include row names and column names and that NA values are currently not supported. In case where only initial estimates of `W` are available, `tca` can be set to re-estimate `W` (see `refit_W`).

`C1` An `n` by `p1` design matrix of covariates that may affect the hidden source-specific values (possibly a different effect on each source). Note that `C1` must include row names and column names and should not include an intercept term. NA values are currently not supported.

`C1.map` An `p1` by `k` matrix of 0/1 values, indicating for each of the `p1` covariates in `C1` whether to consider its potential effects on the values of each of the `k` sources (e.g., if position `i,j` in `C1.map` is 1 then the potential effect of the `i`-th covariate in `C1` on the j-th source will be considered). If `C1.map == NULL` then effects for all covariates in `C1` in each of the sources will be considered.

`C2` An `n` by `p2` design matrix of covariates that may affect the mixture (i.e. rather than directly the sources of the mixture; for example, variables that capture biases in the collection of the measurements). Note that `C2` must include row names and column names and should not include an intercept term. NA values are currently not supported.

`refit_W` A logical value indicating whether to re-estimate the input `W` under the TCA model.

`refit_W.features`
A vector with the names of the features in `X` to consider when re-estimating `W` (i.e. when `refit_W == TRUE`). This is useful since oftentimes just a subset of the features in `X` will be informative for estimating `W`. If `refit_W.features == NULL` then the ReFACTor algorithm will be used for performing feature selection (see also `refit_W.sparsity`,`refit_W.sd_threshold`).

`refit_W.sparsity`
A numeric value indicating the number of features to select using the ReFACTor algorithm when re-estimating `W` (activated only if `refit_W == TRUE` and `refit_W.features == NULL`). Note that `refit_W.sparsity` must be lower or equal to the number of features in `X`. For more information, see the argument `sparsity` in refactor.

`refit_W.sd_threshold`
A numeric value indicating a standard deviation threshold to be used for excluding low-variance features in `X` (activated only if `refit_W == TRUE` and `refit_W.features == NULL`). For more information, see the argument `sd_threshold` in refactor.

`tau` A non-negative numeric value of the standard deviation of the measurement noise (i.e. the i.i.d. component of variation in the model). If `tau == NULL` then `tca` will estimate `tau`.

`parallel` A logical value indicating whether to use parallel computing (possible when using a multi-core machine).

`num_cores` A numeric value indicating the number of cores to use (activated only if `parallel == TRUE`). If `num_cores == NULL` then all available cores except for one will be used.

`max_iters`	A numeric value indicating the maximal number of iterations to use in the optimization of the TCA model (`max_iters` iterations will be used as long as the optimization does not converge in earlier iterations).
`log_file`	A path to an output log file. Note that if the file `log_file` already exists then logs will be appended to the end of the file. Set `log_file` to `NULL` to prevent output from being saved into a file; note that if `verbose == TRUE` then no output file will be generated regardless of the value of `log_file`.
`debug`	A logical value indicating whether to set the logger to a more detailed debug level; please set `debug` to `TRUE` before reporting issues.
`verbose`	A logical value indicating whether to print logs.

Details

The TCA model assumes that the hidden source-specific values are random variables. Formally, denote by Z^i_{hj} the source-specific value of observation i in feature j source h, the TCA model assumes:

$$Z^i_{hj} \sim N(\mu_{hj}, \sigma^2_{hj})$$

where μ_{hj}, σ_{hj} represent the mean and standard deviation that are specific to feature j source h. The model further assumes that the observed value of observation i in feature j is a mixture of k different sources:

$$X_{ji} = \sum_{h=1}^{k} W_{ih} Z^i_{hj} + \epsilon_{ji}$$

where W_{ih} is the non-negative proportion of source h in the mixture of observation i such that $\sum_{h=1}^{k} W_{ih} = 1$, and $\epsilon_{ji} \sim N(0, \tau^2)$ is an i.i.d. component of variation that models measurement noise. Note that the mixture proportions in W are, in general, unique for each individual, therefore each entry in the data matrix X is coming from a unique distribution (i.e. a different mean and a different variance).

In cases where the true `W` is unknown, `tca` can be provided with initial estimates of `W` and then re-estimate `W` as part of the optimization procedure (see argument `refit_W`). These initial estimates should not be random but rather capture the information in `W` to some extent. When the argument `refit_W` is used, it is typically the case that only a subset of the features should be used for re-estimating `W`. Therefore, when re-estimating `W`, `tca` performs feature selection using the ReFACTor algorithm; alternatively, it can also be provided with a user-specified list of features to be used in the re-estimation (see argument `refit_W.features`).

Factors that systematically affect the source-specific values Z^i_{hj} can be further considered (see argument `C1`). In that case, we assume:

$$Z^i_{hj} \sim N(\mu_{hj} + c^{(1)}_i \gamma^h_j, \sigma^2_{hj})$$

where $c^{(1)}_i$ is a row vector from `C1`, corresponding to the values of the p_1 factors for observation i, and γ^h_j is a vector of p_1 corresponding effect sizes.

Factors that systematically affect the mixture values X_{ji}, such as variables that capture biases in the collection of the measurements, can also be considered (see argument `C2`). In that case, we assume:

$$X_{ji} \sim \sum_{h=1}^{k} W_{ih} Z^i_{hj} + c^{(2)}_i \delta_j + \epsilon_{ij}$$

where $c_i^{(2)}$ is a row vector from C2, corresponding to the values of the p_2 factors for observation i, and δ_j is a vector of p_2 corresponding effect sizes.

Value

A list with the estimated parameters of the model. This list can be then used as the input to other functions such as tcareg.

W	An n by k matrix of weights. If refit_W == TRUE then this is the re-estimated W; otherwise this is the input W
mus_hat	An m by k matrix of estimates for the mean of each source in each feature.
sigmas_hat	An m by k matrix of estimates for the standard deviation of each source in each feature.
tau_hat	An estimate of the standard deviation of the i.i.d. component of variation in X. If an input value was provided for tau (i.e. tau != NULL) then tau_hat == tau.
gammas_hat	An m by k*p1 matrix of the estimated effects of the p1 factors in C1 on each of the m features in X, where the first p1 columns are the source-specific effects of the p1 factors on the first source, the following p1 columns are the source-specific effects on the second source and so on.
deltas_hat	An m by p2 matrix of the estimated effects of the p2 factors in C2 on the mixture values of each of the m features in X.

Note

The function tca may require a long running time when the input matrix X is very large; to alleviate this, it is strongly advised to use the parallel argument, given that a multi-core machine is available.

References

Rahmani E, Schweiger R, Rhead B, Criswell LA, Barcellos LF, Eskin E, Rosset S, Sankararaman S, Halperin E. Cell-type-specific resolution epigenetics without the need for cell sorting or single-cell biology. Nature Communications 2018.

Rahmani E, Zaitlen N, Baran Y, Eng C, Hu D, Galanter J, Oh S, Burchard EG, Eskin E, Zou J, Halperin E. Sparse PCA corrects for cell type heterogeneity in epigenome-wide association studies. Nature Methods 2016.

Examples

```
data <- test_data(100, 20, 3, 1, 1, 0.01)
tca.mdl <- tca(X = data$X, W = data$W, C1 = data$C1, C2 = data$C2)
```

tcareg — *Fitting a TCA regression model*

Description

TCA regression allows to test for several types of statistical relations between source-specific values and an outcome of interest. For example, in the context of tissue-level bulk DNA methylation data coming from a mixture of cell types (i.e. the input is methylation sites by individuals), `tcareg` allows to test for cell-type-specific effects of methylation on an outcome of interest.

Usage

```
tcareg(X, tca.mdl, y, C3 = NULL, test = "marginal",
  null_model = NULL, alternative_model = NULL, save_results = TRUE,
  output = "TCA", sort_results = TRUE, parallel = FALSE,
  num_cores = NULL, log_file = "TCA.log", features_metadata = NULL,
  debug = FALSE, verbose = TRUE)
```

Arguments

`X` An `m` by `n` matrix of measurements of `m` features for `n` observations. Each column in `X` is assumed to be a mixture of `k` different sources. Note that `X` must include row names and column names and that NA values are currently not supported. `X` should not include features that are constant across all observations.

`tca.mdl` The value returned by applying the function `tca` to `X`.

`y` An `n` by 1 matrix of an outcome of interest for each of the `n` observations in `X`. Note that `y` must include row names and column names and that NA values are currently not supported.

`C3` An `n` by `p3` design matrix of covariates that may affect `y`. Note that `C3` must include row names and column names and should not include an intercept term. NA values are currently not supported.

`test` A character vector with the type of test to perform on each of the features in `X`; one of the following options: `'marginal'`, `'marginal_conditional'`, `'joint'`, `'single_effect'`, or `'custom'`. Setting `'marginal'` or `'marginal_conditional'` corresponds to testing each feature in `X` for a statistical relation between `y` and each of the `k` sources separately; for any particular source under test, the `marginal_conditional` option further accounts for possible effects of the rest of the `k-1` sources (`'marginal'` will therefore tend to be more powerful in discovering truly related features, but at the same time more prone to falsely tagging the correct related sources if sources are highly correlated). Setting `'joint'` or `'single_effect'` corresponds to testing each feature for an overall statistical relation with `y`, while modeling source-specific effects; the latter option further assumes that the source-specific effects are the same within each feature (`'single_effect'` means only one degree of freedom and will therefore be more powerful when the assumption of a single effect within a feature holds). Finally, `'custom'` corresponds to testing each feature in `X` for a statistical relation with `y` under a user-specified model

(alternative model) with respect to a null model (null model); for example, for testing for relation of the combined (potentially different) effects of features 1 and 2 while accounting for the (potentially different) effects of 3 and 4, set the null model to be sources 3, 4 and the alternative model to be sources 1, 2, 3, 4. Indicating that `null_model` assumes no effect for any of the sources can be done by setting it to `NULL`.

`null_model` A vector with a subset of the names of the sources in `tca.mdl$W` to be used as a null model (activated only if `test == 'custom'`). Note that the null model must be nested within the alternative model; set `null_model` to be `NULL` for indicating no effect for any of the sources under the null model.

`alternative_model` A vector with a subset (or all) of the names of the sources in `tca.mdl$W` to be used as an alternative model (activated only if `test == 'custom'`).

`save_results` A logical value indicating whether to save the returned results in a file. If `TRUE` and `test == 'marginal'` or `test == 'marginal_conditional'` then `k` files will be saved (one for the results of each source).

`output` Prefix for output files (activated only if `save_results == TRUE`).

`sort_results` A logical value indicating whether to sort the results by their p-value (i.e. features with lower p-value will appear first in the results).

`parallel` A logical value indicating whether to use parallel computing (possible when using a multi-core machine).

`num_cores` A numeric value indicating the number of cores to use (activated only if `parallel == TRUE`). If `num_cores == NULL` then all available cores except for one will be used.

`log_file` A path to an output log file. Note that if the file `log_file` already exists then logs will be appended to the end of the file. Set `log_file` to `NULL` to prevent output from being saved into a file; note that if `verbose == TRUE` then no output file will be generated regardless of the value of `log_file`.

`features_metadata` A path to a csv file containing metadata about the features in `X` that will be added to the output files (activated only if `save_results == TRUE`). Each row in the metadata file should correspond to one feature (with the row name being the feature identifier, as it appears in the rows of `X`) and each column should correspond to one metadata descriptor (with an appropriate column name). Features that do not exist in `X` will be ignored and features in `X` with missing metadata information will show missing values.

`debug` A logical value indicating whether to set the logger to a more detailed debug level; please set `debug` to `TRUE` before reporting issues.

`verbose` A logical value indicating whether to print logs.

Details

TCA models Z^i_{hj} as the source-specific value of observation i in feature j coming from source h (see tca for more details). A TCA regression model tests an outcome Y for a linear statistical

relation with the source-specific values of a feature j by assuming:

$$Y_i = \sum_{h=1}^{k} \beta_{hj} Z_{hj}^i + e_i$$

where $e_i \sim N(0, \phi^2)$. In practice, `tcareg` fits this model using the conditional distribution $Y|X$, which, effectively, integrates over the latent Z_{hj}^i parameters. Statistical significance is then calculated using a likelihood ratio test (LRT). Note that the null and alternative models will be set automatically, except when `test == 'custom'`, in which case they will be set according to the user-specified null and alternative hypotheses.

Under the TCA regression model, several statistical tests can be performed by setting the argument `test` according to one of the following options.

1. If `test == 'marginal'`, `tcareg` will perform the following for each source l. For each feature j, β_{lj} will be estimated and tested for a non-zero effect, while assuming $\beta_{hj} = 0$ for all other sources $h \neq l$.

2. If `test == 'marginal_conditional'`, `tcareg` will perform the following for each source l. For each feature j, β_{lj} will be estimated and tested for a non-zero effect, while also estimating the effect sizes β_{hj} for all other sources $h \neq l$.

3. If `test == 'joint'`, `tcareg` will estimate for each feature j the effect sizes of all k sources $\beta_{1j}, \ldots, \beta_{kj}$ and then test the set of k estimates of each feature `j` for a joint effect.

4. If `test == 'single_effect'`, `tcareg` will estimate for each feature j the effect sizes of all k sources $\beta_{1j}, \ldots, \beta_{kj}$, under the assumption that $\beta_{1j} = \ldots = \beta_{kj}$, and then test the set of k estimates of each feature `j` for a joint effect.

5. If `test == 'custom'`, `tcareg` will estimate for each feature j the effect sizes of a predefined set of sources (defined by a user-specified alternative model) and then test their estimates for a joint effect, while accounting for a nested predefined set of sources (defined by a user-specified null model).

Value

A list with the results of applying the TCA regression model to each of the features in the data. If `test == 'marginal'` or `test == 'marginal_conditional'` then a list of `k` such lists of results are returned, one for the results of each source.

`phi`	An estimate of the standard deviation of the i.i.d. component of variation in the TCA regression model.
`beta`	A matrix of effect size estimates for the source-specific effects, such that each row corresponds to the estimated effect sizes of one feature. The number of columns corresponds to the number of estimated effects (e.g., if `test` is set to `marginal` then `beta` will include a single column, if `test` is set to `joint` then `beta` will include `k` columns and so on).
`intercept`	An `m` by `1` matrix of estimates for the intercept of each feature.
`alpha`	An `m` by `p3` matrix of effect size estimates for the `p3` covariates in `C3`, such that each row corresponds to the estimated effect sizes of one feature.
`null_ll`	An `m` by `1` matrix of the log-likelihood of the model under the null hypothesis.

alternative_ll	An m by 1 matrix of the log-likelihood of the model under the alternative hypothesis.
stats	An m by 1 matrix of the LRT statistic for each feature in the data.
df	The degrees of freedom for deriving p-values using LRT.
pvals	An m by 1 matrix of the p-value for each feature in the data.
qvals	An m by 1 matrix of the q-value (FDR-adjusted p-values) for each feature in the data.

Note

The function tcareg may require a long running time when the input matrix X is very large; to alleviate this, it is strongly advised to use the parallel argument, given that a multi-core machine is available.

References

Rahmani E, Schweiger R, Rhead B, Criswell LA, Barcellos LF, Eskin E, Rosset S, Sankararaman S, Halperin E. Cell-type-specific resolution epigenetics without the need for cell sorting or single-cell biology. Nature Communications 2018.

Examples

```
n <- 50
m <- 10
k <- 3
p1 <- 1
p2 <- 1
data <- test_data(n, m, k, p1, p2, 0.01)
tca.mdl <- tca(X = data$X, W = data$W, C1 = data$C1, C2 = data$C2)
y <- matrix(rexp(n, rate=.1), ncol=1)
# joint test:
res1 <- tcareg(data$X, tca.mdl, y, test = "joint", save_results = FALSE)
# custom test, testing for a joint effect of sources 1,2 while accounting for source 3
res2 <- tcareg(data$X, tca.mdl, y, test = "custom", null_model = c("3"),
alternative_model = c("1","2","3"), save_results = FALSE)
# custom test, testing for a joint effect of sources 1,2 assuming no effects under the null
res3 <- tcareg(data$X, tca.mdl, y, test = "custom", null_model = NULL,
alternative_model = c("1","2"), save_results = FALSE)
```

---

tcasub *Subsetting features from a TCA model*

Description

Extracts from a fitted TCA model (i.e. a value returned by the function tca) a subset of the features.

Usage

```
tcasub(tca.mdl, features, log_file = "TCA.log", debug = FALSE,
  verbose = TRUE)
```

Arguments

tca.mdl	The value returned by applying the function tca to some data matrixX.
features	A vector with the identifiers of the features to extract (as they appear in the rows of X).
log_file	A path to an output log file. Note that if the file log_file already exists then logs will be appended to the end of the file. Set log_file to NULL to prevent output from being saved into a file; note that if verbose == TRUE then no output file will be generated regardless of the value of log_file.
debug	A logical value indicating whether to set the logger to a more detailed debug level; please set debug to TRUE before reporting issues.
verbose	A logical value indicating whether to print logs.

Details

This function allows to extract a subset of the features from a fitted TCA model (i.e. from a value returned by the function tca). This allows, for example, to extract and then perform post-hoc tests on only a small set of candidate features (e.g., using the function tcareg), without the need to run tca again for fitting the model to the candidate features.

Value

A list with the estimated parameters of the model for the given set of features.

W	Equals to tca.mdl$W
mus_hat	A q by k matrix which is a subset of the matrix tca.mdl$mus_hat, where q is the number of features in the argument features.
sigmas_hat	A q by k matrix which is a subset of the matrix tca.mdl$sigmas_hat, where q is the number of features in the argument features.
tau_hat	Equals to tca.mdl$tau_hat
gammas_hat	A q by k*p1 matrix which is a subset of the matrix tca.mdl$gammas_hat, where q is the number of features in the argument features.
deltas_hat	A q by p2 matrix which is a subset of the matrix tca.mdl$deltas_hat, where q is the number of features in the argument features.

Examples

```
data <- test_data(50, 20, 3, 0, 0, 0.01)
tca.mdl <- tca(X = data$X, W = data$W)
tca.mdl.subset <- tcasub(tca.mdl, rownames(data$X)[1:10])
y <- matrix(rexp(50, rate=.1), ncol=1)
# run tcareg test with an outcome y:
res <- tcareg(data$X[1:10,], tca.mdl.subset, y, test = "joint", save_results = FALSE)
```

`tensor` *Extracting hidden 3D signals from 2D input*

Description

Estimates 3-dimensional signals (features by observations by sources) from input of mixtures (features by observations), under the assumption of the TCA model that each observation is a mixture of unique source-specific values (in each feature in the data). For example, in the context of tissue-level bulk DNA methylation data coming from a mixture of cell types (i.e. the input is methylation sites by individuals), `tensor` allows to estimate a tensor of cell-type-specific levels for each individual in each methylation site (i.e. a tensor of methylation sites by individuals by cell types).

Usage

```
tensor(X, tca.mdl, parallel = FALSE, num_cores = NULL,
  log_file = "TCA.log", debug = FALSE, verbose = TRUE)
```

Arguments

`X`	An `m` by `n` matrix of measurements of `m` features for `n` observations. Each column in `X` is assumed to be a mixture of `k` different sources. Note that `X` must include row names and column names and that NA values are currently not supported. `X` should not include features that are constant across all observations.
`tca.mdl`	The value returned by applying the function `tca` to `X`.
`parallel`	A logical value indicating whether to use parallel computing (possible when using a multi-core machine).
`num_cores`	A numeric value indicating the number of cores to use (activated only if `parallel == TRUE`). If `num_cores == NULL` then all available cores except for one will be used.
`log_file`	A path to an output log file. Note that if the file `log_file` already exists then logs will be appended to the end of the file. Set `log_file` to `NULL` to prevent output from being saved into a file; note that if `verbose == TRUE` then no output file will be generated regardless of the value of `log_file`.
`debug`	A logical value indicating whether to set the logger to a more detailed debug level; please set `debug` to `TRUE` before reporting issues.
`verbose`	A logical value indicating whether to print logs.

Details

See tca for notations and details about the TCA model. Given estimates of the parameters of the model (given by tca), `tensor` uses the conditional distribution $Z^i_{hj}|X_{ji}$ for estimating the k source-specific levels of each observation i in each feature j.

Value

A list with the estimated source-specific values. The first element in the list is an `m` by `n` matrix (features by observations) corresponding to the estimated values coming from the first source, the second element in the list is another `m` by `n` matrix (features by observations) corresponding to the estimated values coming from the second source and so on.

References

Rahmani E, Schweiger R, Rhead B, Criswell LA, Barcellos LF, Eskin E, Rosset S, Sankararaman S, Halperin E. Cell-type-specific resolution epigenetics without the need for cell sorting or single-cell biology. Nature Communications 2018.

Examples

```
data <- test_data(50, 20, 3, 2, 2, 0.01)
tca.mdl <- tca(X = data$X, W = data$W, C1 = data$C1, C2 = data$C2)
Z_hat <- tensor(data$X, tca.mdl)
```

`test_data` *Generate test data*

Description

Generates simple test data following the TCA model.

Usage

```
test_data(n, m, k, p1, p2, tau, log_file = "TCA.log", verbose = TRUE)
```

Arguments

`n`	The number of observations to simulate.
`m`	The number of features to simulate.
`k`	The number of sources to simulate.
`p1`	The number of covariates that affect the source-specific values to simulate.
`p2`	The number of covariates that affect the mixture values to simulate.
`tau`	The variance of the i.i.d. component of variation to add on top of the simulated mixture values.
`log_file`	A path to an output log file. Note that if the file `log_file` already exists then logs will be appended to the end of the file. Set `log_file` to `NULL` to prevent output from being saved into a file; note that if `verbose == TRUE` then no output file will be generated regardless of the value of `log_file`.
`verbose`	A logical value indicating whether to print logs.

Details

See tca for details about the TCA model.

Value

A list with the simulated data and parameters.

`X`	An `m` by `n` matrix of simulated data with `m` features for `n` observations.
`Z`	A list with the simulated source-specific values, where the first element in the list is an `m` by `n` matrix (features by observations) corresponding to the values coming from the first source, the second element in the list is another `m` by `n` matrix (features by observations) corresponding to the values coming from the second source and so on.
`W`	An `n` by `k` matrix of simulated weights - the weights of the `k` sources for each of the `n` mixtures (observations).
`mus`	An `m` by `k` matrix of the mean of each of the `m` features for each of the `k` sources.
`sigmas`	An `m` by `k` matrix of the standard variation of each of the `m` features for each of the `k` sources.
`C1`	An `n` by `p1` design matrix of simulated covariates that affect the hidden source-specific values.
`C2`	An `n` by `p2` design matrix of simulated covariates that affect the mixture.
`gammas`	An `m` by `k*p1` matrix of the effects of the `p1` factors in `C1` on each of the `m` features in `X`, where the first `p1` columns are the source-specific effects of the `p1` factors on the first source, the following `p1` columns are the source-specific effects on the second source and so on.
`deltas`	An `m` by `p2` matrix of the effects of the `p2` factors in `C2` on the mixture values of each of the `m` features in `X`.

Examples

```
data <- test_data(100, 50, 3, 2, 2, 0.01)
```

Index

refactor, 2, *5*

tca, 4, *9*, *13*, *15*
tcareg, 8
tcasub, 11
tensor, 13
test_data, 14