Package ‘coseq’

May 29, 2024

Type Package
Title Co-Expression Analysis of Sequencing Data
Version 1.28.0
Date 2021-07-21
Depends R (>= 4.0.0), SummarizedExperiment, S4Vectors
Imports edgeR, DESeq2, capushe, Rmixmod, e1071, BiocParallel, ggplot2,
    scales, HTSFILTER, corplot, HTS Cluser, grDevices, graphics,
    stats, methods, compositions, mvtnorm
Suggests Biobase, knitr, rmarkdown, testthat, BiocStyle
Description Co-expression analysis for expression profiles arising from
    high-throughput sequencing data. Feature (e.g., gene) profiles
    are clustered using adapted transformations and mixture models or
    a K-means algorithm, and model selection criteria
    (to choose an appropriate number of clusters) are provided.
biocViews GeneExpression, RNASeq, Sequencing, Software, ImmunoOnco
License GPL-3
LazyLoad yes
Encoding UTF-8
RoxygenNote 7.1.1
VignetteBuilder knitr
git_url https://git.bioconductor.org/packages/coseq
git_branch RELEASE_3_19
git_last_commit 9fd4a0b9
git_last_commit_date 2024-04-30
Repository Bioconductor 3.19
Date/Publication 2024-05-29
Author Andrea Rau [cre, aut] (<https://orcid.org/0000-0001-6469-488X>),
    Cathy Maugis-Rabusseau [ctb],
    Antoine Godichon-Baggioni [ctb]
Maintainer Andrea Rau <andrea.rau@inrae.fr>
Description

Co-expression analysis for expression profiles arising from high-throughput sequencing data. Feature (e.g., gene) profiles are clustered using adapted transformations and mixture models or a K-means algorithm, and model selection criteria (to choose an appropriate number of clusters) are provided.

Details

<table>
<thead>
<tr>
<th>Package:</th>
<th>coseq</th>
</tr>
</thead>
<tbody>
<tr>
<td>Type:</td>
<td>Package</td>
</tr>
<tr>
<td>Version:</td>
<td>1.15.4</td>
</tr>
<tr>
<td>Date:</td>
<td>2020-12-03</td>
</tr>
<tr>
<td>License:</td>
<td>GPL-3</td>
</tr>
<tr>
<td>LazyLoad:</td>
<td>yes</td>
</tr>
</tbody>
</table>
Author(s)
Andrea Rau, Cathy Maugis-Rabusseau, Antoine Godichon-Baggioni
Maintainer: Andrea Rau <andrea.rau@inrae.fr>

References

Examples
```r
## Simulate toy data, n = 300 observations
set.seed(12345)
countmat <- matrix(runif(300*4, min=0, max=500), nrow=300, ncol=4)
countmat <- countmat[which(rowSums(countmat) > 0),]
conds <- rep(c("A","B","C","D"), each=2)

## Run the Normal mixture model for K = 2,3,4
run_arcsin <- coseq(object=countmat, K=2:4, iter=5, transformation="arcsin",
                   model="Normal", seed=12345)
run_arcsin

## Plot and summarize results
plot(run_arcsin)
summary(run_arcsin)

## Compare ARI values for all models (no plot generated here)
ARI <- compareARI(run_arcsin, plot=FALSE)

## Compare ICL values for models with arcsin and logit transformations
run_logit <- coseq(object=countmat, K=2:4, iter=5, transformation="logit",
                   model="Normal")
compareICL(list(run_arcsin, run_logit))

## Use accessor functions to explore results
clusters(run_arcsin)
likelihood(run_arcsin)
nbCluster(run_arcsin)
ICL(run_arcsin)

## Examine transformed counts and profiles used for graphing
```

## Run the K-means algorithm for logclr profiles for K = 2,..., 20

```r
run_kmeans <- coseq(object=countmat, K=2:20, transformation="logclr",
                     model="kmeans")
run_kmeans
```

### clusterEntropy

**Calculation of per-cluster entropy**

**Description**

Provides the calculation of per-cluster entropy, equivalent to

\[
\text{Entropy}(k) = \sum_{i \in C_k} \log(\tau_{ik})
\]

where \(\tau_{ik}\) is the conditional probability of gene \(i\) belonging to cluster \(k\) and \(C_k\) corresponds to the set of indices of genes attributed to cluster \(k\).

**Usage**

```r
clusterEntropy(probaPost)
```

**Arguments**

- `probaPost` Matrix containing the conditional probabilities of belonging to each cluster for all observations

**Value**

Entropy per cluster

**Author(s)**

Cathy Maugis-Rabusseau

**Examples**

```r
## Generate artificial matrix of conditional probabilities for K=5 clusters
tmp <- matrix(runif(100*5), nrow=100, ncol=5)
probaPost <- tmp / rowSums(tmp)
clusterEntropy(probaPost)
```
clusterInertia

Description

Provides the calculation of within-cluster inertia, equivalent to

\[ \text{Inertia}(k) = \sum_{i \in C_k} (y_{ik} - \mu_k)^2 \]

where \( \mu_k \) is the mean of cluster \( k \) and \( C_k \) corresponds to the set of indices of genes attributed to cluster \( k \).

Usage

clusterInertia(profiles, clusters)

Arguments

profiles Matrix, data.frame, or DataFrame containing the (transformed) profiles used for the clustering
clusters Vector of cluster labels corresponding to the observations in profiles

Value

Within cluster inertia

Author(s)

Andrea Rau, Antoine Godichon-Baggioni

Examples

```r
## Simulate toy data, n = 300 observations
set.seed(12345)
countmat <- matrix(runif(300*4, min=0, max=500), nrow=300, ncol=4)
countmat <- countmat[which(rowSums(countmat) > 0),]
conds <- rep(c(“A”,”B”,”C”,”D”), each=2)

## Run the K-means algorithm for logclr profiles for K = 2,..., 20
run_kmeans <- coseq(object=countmat, K=2:20, transformation=”logclr”,
                    model=”kmeans”)
countmat <- t(countmat)
countmat <- t(countmat)
clusterInertia(profiles=tcounts(run_kmeans), clusters=clusters(run_kmeans))
```
compareARI

Pairwise comparisons of ARI values among a set of clustering partitions

Description

Provides the adjusted rand index (ARI) between pairs of clustering partitions.

Usage

```r
compareARI(object, ...)  
## S4 method for signature 'coseqResults'
compareARI(
  object,
  K = NULL,
  parallel = FALSE,
  BPPARAM = bpparam(),
  plot = TRUE,
  ...
)
## S4 method for signature 'matrix'
compareARI(object, parallel = FALSE, BPPARAM = bpparam(), plot = TRUE, ...)
## S4 method for signature 'data.frame'
compareARI(object, parallel = FALSE, BPPARAM = bpparam(), plot = TRUE, ...)
```

Arguments

- `object`: Object of class coseqResults or RangedSummarizedExperiment, or alternatively a n x M data.frame or matrix containing the clustering partitions for M different models.
- `...`: Additional optional parameters for corrplot.
- `K`: If NULL, pairwise ARI values will be calculated among every model in object x. Otherwise, K provides a vector of cluster numbers identifying a subset of models in x.
- `parallel`: If FALSE, no parallelization. If TRUE, parallel execution using BiocParallel (see next argument BPPARAM). Note that parallelization is unlikely to be helpful unless the number of observations n in the clustering partitions or the number of models M are very large.
- `BPPARAM`: Optional parameter object passed internally to bplapply when parallel=TRUE. If not specified, the parameters last registered with register will be used.
- `plot`: If TRUE, provide a heatmap using corrplot to visualize the calculated pairwise ARI values.
**Value**

Matrix of adjusted rand index values calculated between each pair of models.

**Author(s)**

Andrea Rau

**Examples**

```r
## Simulate toy data, n = 300 observations
set.seed(12345)
countmat <- matrix(runif(300*4, min=0, max=500), nrow=300, ncol=4)
countmat <- countmat[which(rowSums(countmat) > 0),]
conds <- rep(c("A", "B", "C", "D"), each=2)

## Run the Normal mixture model for K = 2,3,4
run_arcsin <- coseq(object=countmat, K=2:4, iter=5, transformation="arcsin",
                     model="Normal", seed=12345)
run_arcsin

## Plot and summarize results
plot(run_arcsin)
summary(run_arcsin)

## Compare ARI values for all models (no plot generated here)
ARI <- compareARI(run_arcsin, plot=FALSE)

## Compare ICL values for models with arcsin and logit transformations
run_logit <- coseq(object=countmat, K=2:4, iter=5, transformation="logit",
                   model="Normal")
compareICL(list(run_arcsin, run_logit))

## Use accessor functions to explore results
clusters(run_arcsin)
likelihood(run_arcsin)
nbCluster(run_arcsin)
ICL(run_arcsin)

## Examine transformed counts and profiles used for graphing
tcounts(run_arcsin)
profiles(run_arcsin)

## Run the K-means algorithm for logclr profiles for K = 2,..., 20
run_kmeans <- coseq(object=countmat, K=2:20, transformation="logclr",
                   model="kmeans")
run_kmeans
```
**compareICL**

*Compare corrected ICL values after data transformation*

**Description**

Compare the corrected ICL values after applying the arcsin, logit, and logMedianRef transformations in a coseq analysis.

**Usage**

```r
compareICL(x)
```

**Arguments**

- `x`: A list made up of `coseqResults` objects. At the current time, this function only supports the comparison of `coseqResults` objects using `model="Normal"` and `transformation = c("arcsin", "logit", "logMedianRef")`.

**Value**

A plot of corrected ICL values for the models included in `x` (the list of `coseqResults` objects).

**Author(s)**

Andrea Rau, Cathy Maugis-Rabusseau

**Examples**

```r
## Simulate toy data, n = 300 observations
set.seed(12345)
countmat <- matrix(runif(300*4, min=0, max=500), nrow=300, ncol=4)
countmat <- countmat[which(rowSums(countmat) > 0),]
conds <- rep(c("A", "B", "C", "D"), each=2)

## Run the Normal mixture model for K = 2,3,4
run_arcsin <- coseq(object=countmat, K=2:4, iter=5, transformation="arcsin",
                    model="Normal", seed=12345)
run_arcsin

## Plot and summarize results
plot(run_arcsin)
summary(run_arcsin)

## Compare ARI values for all models (no plot generated here)
ARI <- compareARI(run_arcsin, plot=FALSE)

## Compare ICL values for models with arcsin and logit transformations
run_logit <- coseq(object=countmat, K=2:4, iter=5, transformation="logit",
                   model="Normal")
run_logit
```
compareICL(list(run_arcsin, run_logit))

## Use accessor functions to explore results
clusters(run_arcsin)
likelihood(run_arcsin)
nbCluster(run_arcsin)
ICL(run_arcsin)

## Examine transformed counts and profiles used for graphing
tcounts(run_arcsin)
profiles(run_arcsin)

## Run the K-means algorithm for logclr profiles for K = 2,..., 20
run_kmeans <- coseq(object=countmat, K=2:20, transformation="logclr", model="kmeans")
run_kmeans

---

**convertLegacyCoseq**  
*Convert legacy coseq objects*

**Description**

Convert legacy coseq S3 class objects to coseqResults S4 class objects

**Usage**

convertLegacyCoseq(object, digits = 3)

**Arguments**

- **object**  
  Object of S3 class coseq arising from a call to previous versions of coseq (< 0.99.1)

- **digits**  
  integer indicating the number of decimal places (round) to retain in results.

**Value**

Converted object of S4 class coseqResults compatible with recent versions of coseq (>= 0.99.1)
Co-expression or co-abudance analysis of high-throughput sequencing data

Description

This is the primary user interface for the coseq package. Generic S4 methods are implemented to perform co-expression or co-abudance analysis of high-throughput sequencing data, with or without data transformation, using K-means or mixture models. The supported classes are matrix, data.frame, and DESeqDataSet. The output of coseq is an S4 object of class coseqResults.

Usage

coseq(object, ...)

## S4 method for signature 'matrix'
coseq(
  object,
  K,
  subset = NULL,
  model = "kmeans",
  transformation = "logclr",
  normFactors = "TMM",
  meanFilterCutoff = NULL,
  modelChoice = ifelse(model == "kmeans", "DDSE", "ICL"),
  parallel = FALSE,
  BPPARAM = bpparam(),
  seed = NULL,
  ...
)

## S4 method for signature 'data.frame'
coseq(
  object,
  K,
  subset = NULL,
  model = "kmeans",
  transformation = "logclr",
  normFactors = "TMM",
  meanFilterCutoff = NULL,
  modelChoice = ifelse(model == "kmeans", "DDSE", "ICL"),
  parallel = FALSE,
  BPPARAM = bpparam(),
  seed = NULL,
  ...
)
coseq

## S4 method for signature 'DESeqDataSet'

```r
coseq(
  object,
  K,
  model = "kmeans",
  transformation = "logclr",
  normFactors = "TMM",
  meanFilterCutoff = NULL,
  modelChoice = ifelse(model == "kmeans", "DDSE", "ICL"),
  parallel = FALSE,
  BPPARAM = bpparam(),
  seed = NULL,
  ...
)
```

### Arguments

- **object**: Data to be clustered. May be provided as a \( n \times q \) matrix or data.frame of observed counts for \( n \) observations and \( q \) variables, or an object of class `DESeqDataSet` arising from a differential analysis via DESeq2.
- **K**: Number of clusters (a single value or a vector of values).
- **subset**: Optional vector providing the indices of a subset of genes that should be used for the co-expression analysis (i.e., row indices of the data matrix \( y \). For the generic function `coseq`, the results of a previously run differential analysis may be used to select a subset of genes on which to perform the co-expression analysis. If this is desired, `subset.index` can also be an object of class `DESeqResults` (from the `results` function in DESeq2).
- **model**: Type of mixture model to use ("Poisson" or "Normal"), or alternatively "kmeans" for a K-means algorithm.
- **transformation**: Transformation type to be used: "voom", "logRPKM" (if geneLength is provided by user), "arcsin", "logit", "logMedianRef", "profile", "logclr", "clr", "alr", "ilr", or "none".
- **normFactors**: The type of estimator to be used to normalize for differences in library size: ("TC" for total count, "UQ" for upper quantile, "Med" for median, "DESeq" for the normalization method in the DESeq package, and "TMM" for the TMM normalization method (Robinson and Oshlack, 2010). Can also be a vector (of length \( q \)) containing pre-estimated library size estimates for each sample, or "none" if no normalization is required.
- **meanFilterCutoff**: Value used to filter low mean normalized counts if desired (by default, set to a value of 50).
- **modelChoice**: Criterion used to select the best model. For Gaussian mixture models, "ICL" (integrated completed likelihood criterion) is currently supported. For Poisson mixture models, "ICL", "BIC" (Bayesian information criterion), and a non-asymptotic criterion calibrated via the slope heuristics using either the "DDSE"
(data-driven slope estimation) or “Djump” (dimension jump) approaches may be used. See the HTSCluster package documentation for more details about the slope heuristics approaches.

**parallel**

If FALSE, no parallelization. If TRUE, parallel execution using BiocParallel (see next argument BPPARAM). A note on running in parallel using BiocParallel: it may be advantageous to remove large, unneeded objects from the current R environment before calling the function, as it is possible that R's internal garbage collection will copy these files while running on worker nodes.

**BPPARAM**

Optional parameter object passed internally to `bplapply` when parallel=TRUE. If not specified, the parameters last registered with `register` will be used.

**seed**

If desired, an integer defining the seed of the random number generator. If NULL, a random seed is used.

**Value**

An S4 object of class `coseqResults`, where conditional probabilities of cluster membership for each gene in each model is stored as a SimpleList of assay data, and the corresponding log likelihood, ICL value, number of clusters, and form of Gaussian model for each model are stored as metadata.

**Author(s)**

Andrea Rau

**Examples**

```r
## Simulate toy data, n = 300 observations
set.seed(12345)
countmat <- matrix(runif(300*4, min=0, max=500), nrow=300, ncol=4)
countmat <- countmat[which(rowSums(countmat) > 0),]
conds <- rep(c("A", "B", "C", "D"), each=2)

## Run the Normal mixture model for K = 2,3,4
run_arcsin <- coseq(object=countmat, K=2:4, iter=5, transformation="arcsin", model="Normal", seed=12345)
run_arcsin

## Plot and summarize results
plot(run_arcsin)
summary(run_arcsin)

## Compare ARI values for all models (no plot generated here)
ARI <- compareARI(run_arcsin, plot=FALSE)

## Compare ICL values for models with arcsin and logit transformations
run_logit <- coseq(object=countmat, K=2:4, iter=5, transformation="logit", model="Normal")
compareICL(list(run_arcsin, run_logit))

## Use accessor functions to explore results
```
coseqFullResults

clusters(run_arcsin)
likelihood(run_arcsin)
nbCluster(run_arcsin)
ICL(run_arcsin)

## Examine transformed counts and profiles used for graphing
tcounts(run_arcsin)
profiles(run_arcsin)

## Run the K-means algorithm for logclr profiles for K = 2,..., 20
run_kmeans <- coseq(object=countmat, K=2:20, transformation="logclr",
model="kmeans")
run_kmeans

---

coseqFullResults  
Accessors for the assigned cluster labels of a coseqResults object.

Description

The counts slot holds the count data as a matrix of non-negative integer count values, one row for each observational unit (gene or the like), and one column for each sample.

Usage

coseqFullResults(object, ...)
clusters(object, ...)
likelihood(object, ...)
nbCluster(object, ...)
proba(object, ...)
ICL(object, ...)
profiles(object, ...)
tcounts(object, ...)
transformationType(object, ...)
model(object, ...)
DDSEextract(object, ...)
Djumpextract(object, ...)

## S4 method for signature 'coseqResults'
clusters(object, K)

## S4 method for signature 'RangedSummarizedExperiment'
clusters(object, ...)

## S4 method for signature 'matrix'
clusters(object, ...)

## S4 method for signature 'data.frame'
clusters(object, ...)

## S4 method for signature 'MixmodCluster'
likelihood(object)

## S4 method for signature 'RangedSummarizedExperiment'
likelihood(object)

## S4 method for signature 'coseqResults'
likelihood(object)

## S4 method for signature 'NULL'
likelihood(object)

## S4 method for signature 'MixmodCluster'
nbCluster(object)

## S4 method for signature 'RangedSummarizedExperiment'
nbCluster(object)

## S4 method for signature 'coseqResults'
nbCluster(object)

## S4 method for signature 'NULL'
nbCluster(object)

## S4 method for signature 'MixmodCluster'
ICL(object)

## S4 method for signature 'RangedSummarizedExperiment'
ICL(object)

## S4 method for signature 'coseqResults'
ICL(object)

## S4 method for signature 'NULL'
ICL(object)
## S4 method for signature 'coseqResults'
profiles(object)

## S4 method for signature 'coseqResults'
tcounts(object)

## S4 method for signature 'coseqResults'
transformationType(object)

## S4 method for signature 'coseqResults'
model(object)

## S4 method for signature 'coseqResults'
coseqFullResults(object)

## S4 method for signature 'coseqResults'
show(object)

## S4 method for signature 'MixmodCluster'
proba(object)

## S4 method for signature 'Capushe'
DDSEextract(object)

## S4 method for signature 'Capushe'
Djumpextract(object)

### Arguments

- **object**: a `coseqResults`, `RangedSummarizedExperiment`, or `MixmodCluster` object.
- **...**: Additional optional parameters
- **K**: numeric indicating the model to be used (if NULL or missing, the model chosen by ICL is used by default)

### Value

Output varies depending on the method. `clusters` returns a vector of cluster labels for each gene for the desired model.

### Author(s)

Andrea Rau

### Examples

```r
## Simulate toy data, n = 300 observations
set.seed(12345)
countmat <- matrix(runif(300*4, min=0, max=500), nrow=300, ncol=4)
countmat <- countmat[which(rowSums(countmat) > 0),]
```
conds <- rep(c("A","B","C","D"), each=2)

## Run the Normal mixture model for K = 2,3,4
run_arcsin <- coseq(object=countmat, K=2:4, iter=5, transformation="arcsin",
                   model="Normal", seed=12345)
run_arcsin

## Plot and summarize results
plot(run_arcsin)
summary(run_arcsin)

## Compare ARI values for all models (no plot generated here)
ARI <- compareARI(run_arcsin, plot=FALSE)

## Compare ICL values for models with arcsin and logit transformations
run_logit <- coseq(object=countmat, K=2:4, iter=5, transformation="logit",
                   model="Normal")
compareICL(list(run_arcsin, run_logit))

## Use accessor functions to explore results
clusters(run_arcsin)
likelihood(run_arcsin)
betweenCluster(run_arcsin)
ICL(run_arcsin)

## Examine transformed counts and profiles used for graphing
tcounts(run_arcsin)
profiles(run_arcsin)

## Run the K-means algorithm for logclr profiles for K = 2,..., 20
run_kmeans <- coseq(object=countmat, K=2:20, transformation="logclr",
                   model="kmeans")
run_kmeans

---

coseqResults-class  coseqResults object and constructor

**Description**

coseqResults is a subclass of RangedSummarizedExperiment, used to store the co-expression
results as well as some additional information useful for plotting (tcounts, y_profiles) and meta-
information about the co-expression analysis (transformation, normFactors).

**Usage**

coseqResults(
  SummarizedExperiment,
  allResults,
  model = NULL,
  transformation = NULL,
Arguments

- `SummarizedExperiment`: a `RangedSummarizedExperiment` of coseq results
- `allResults`: List of conditional probabilities of cluster membership for each gene, in all models fit
- `model`: "Normal" or "Poisson", the mixture model used for co-expression
- `transformation`: Transformation applied to counts to obtain `tcounts`
- `tcounts`: Transformed counts used for mixture model fitting
- `y_profiles`: y profiles used for coseq plotting
- `normFactors`: Scaling factors used for normalization

Details

This constructor function would not typically be used by "end users". This simple class extends the `RangedSummarizedExperiment` class of the `SummarizedExperiment` package to allow other packages to write methods for results objects from the coseq package. It is used by `coseqRun` to wrap up the results table.

Value

- a `coseqResults` object

Description

Function for primary code to perform co-expression analysis, with or without data transformation, using mixture models. The output of `coseqRun` is an S4 object of class `coseqResults`.

Usage

```r
coseqRun(
  y,
  K,
  conds = NULL,
  normFactors = "TMM",
  model = "kmeans",
  transformation = "logclr",
  subset = NULL,
)
```
meanFilterCutoff = 50,
modelChoice = ifelse(model == "kmeans", "DDSE", "ICL"),
parallel = FALSE,
BPPARAM = bpparam(),
seed = NULL,
...)

Arguments

y 
(n x q) matrix of observed counts for n observations (genes) and q variables (samples). In nearly all cases, n > q.

K 
Number of clusters (a single value or a vector of values)

conds 
Vector of length q defining the condition (treatment group) for each variable (column) in y

normFactors 
The type of estimator to be used to normalize for differences in library size: ("TC" for total count, "UQ" for upper quantile, "Med" for median, "DESeq" for the normalization method in the DESeq package, and "TMM" for the TMM normalization method (Robinson and Oshlack, 2010). Can also be a vector (of length q) containing pre-estimated library size estimates for each sample, or "none" if no normalization is required.

model 
Type of mixture model to use ("Poisson" or "Normal"), or alternatively "kmeans" for a K-means algorithm

transformation 
Transformation type to be used: "voom", "logRPKM" (if geneLength is provided by user), "arcsin", "logit", "logMedianRef", "profile", "logclr", "clr", "alr", "ilr", or "none"

subset 
Optional vector providing the indices of a subset of genes that should be used for the co-expression analysis (i.e., row indices of the data matrix y. For the generic function coseq, the results of a previously run differential analysis may be used to select a subset of genes on which to perform the co-expression analysis. If this is desired, subset.index can also be an object of class DESeqResults (from the results function in DESeq2).

meanFilterCutoff 
Value used to filter low mean normalized counts if desired (by default, set to a value of 50)

modelChoice 
Criterion used to select the best model. For Gaussian mixture models, "ICL" (integrated completed likelihood criterion) is currently supported. For Poisson mixture models, "ICL", "BIC" (Bayesian information criterion), and a non-asymptotic criterion calibrated via the slope heuristics using either the "DDSE" (data-driven slope estimation) or "Djump" (dimension jump) approaches may be used. See the HTSCluster package documentation for more details about the slope heuristics approaches.

parallel 
If FALSE, no parallelization. If TRUE, parallel execution using BiocParallel (see next argument BPPARAM). A note on running in parallel using BiocParallel: it may be advantageous to remove large, unneeded objects from the current R environment before calling the function, as it is possible that R’s internal garbage collection will copy these files while running on worker nodes.
BPPARAM Optional parameter object passed internally to bplapply when parallel=TRUE. If not specified, the parameters last registered with register will be used.

seed If desired, an integer defining the seed of the random number generator. If NULL, a random seed is used.

... Additional optional parameters.

Value

An S4 object of class coseqResults whose assays contain a SimpleList object, where each element in the list corresponds to the conditional probabilities of cluster membership for each gene in each model. Meta data (accessible via metadata include the model used (either Normal or Poisson), the transformation used on the data, the transformed data used to estimate model (tcounts), the normalized profiles for use in plotting (y_profiles), and the normalization factors used in the analysis (normFactors).

Author(s)

Andrea Rau

Examples

```r
## Simulate toy data, n = 300 observations
set.seed(12345)
countmat <- matrix(runif(300*4, min=0, max=500), nrow=300, ncol=4)
countmat <- countmat[which(rowSums(countmat) > 0),]
conds <- rep(c("A","B","C","D"), each=2)

## Run the K-means for K = 2,3,4 with logCLR transformation
## The following are equivalent:
run <- coseqRun(y=countmat, K=2:15)
run <- coseq(object=countmat, K=2:15, transformation="logclr", model="kmeans")

## Run the Normal mixture model for K = 2,3,4 with arcsine transformation
## The following are equivalent:
run <- coseqRun(y=countmat, K=2:4, iter=5, transformation="arcsin", model="Normal")
run <- coseq(object=countmat, K=2:4, iter=5, transformation="arcsin", model="Normal")
```

Description

This dataset represents RNA-seq data from mouse neocortex in Fietz et al. (2012).
Usage

data(fietz)

Format

An ExpressionSet named fietz.eset containing the phenotype data and expression data for the Fietz et al. (2012) experiment. Phenotype data may be accessed using the pData function, and expression data may be accessed using the exprs function.

Value

Object of class ‘ExpressionSet’. Matrix of counts can be accessed after loading the ‘Biobase’ package and calling exprs(fietz)).

Source

Digital Expression Explorer (http://dee.bakeridi.edu.au/).

References

https://perso.math.univ-toulouse.fr/maugis/mixstatseq/packages


kmeansProbaPost

Calculate conditional probabilities of cluster membership for K-means clustering

Description

Calculate conditional probabilities of cluster membership for K-means clustering

Usage

kmeansProbaPost(clusters, tcounts)

Arguments

clusters Cluster labels arising from K-means clustering
tcounts Transformed counts clustered using K-means

Value

Conditional probabilities of cluster membership for each observation in each cluster
### Examples

```r
## Example of K-means taken from ?kmeans help page
x <- rbind(matrix(rnorm(100, sd = 0.3), ncol = 2),
           matrix(rnorm(100, mean = 1, sd = 0.3), ncol = 2))
colnames(x) <- c("x", "y")
cl <- kmeans(x, 5)
probaPost <- kmeansProbaPost(cl$cluster, x)
head(probaPost)
```

---

**logclr**

*Calculate the Log Centered Log Ratio (logCLR) transformation*

#### Description

Calculate the Log Centered Log Ratio (logCLR) transformation

#### Usage

```r
logclr(profiles)
```

#### Arguments

- `profiles` Matrix of profiles. Note that the presence of 0 values causes an error message to be produced.

#### Value

logCLR-transformed profiles

---

**matchContTable**

*Permute columns of a contingency table*

#### Description

Permute the columns of a contingency table comparing two clusterings to load the diagonal as much as possible.

#### Usage

```r
matchContTable(table_1, table_2)
```

#### Arguments

- `table_1` Partition from a first data clustering
- `table_2` Partition from a second data clustering
Value

Permuted contingency table

Examples

```r
## Generate arbitrary labels from two separate clustering results
labels_1 <- sample(1:10, 1000, replace=TRUE)  ## K=10 clusters
labels_2 <- sample(1:8, 1000, replace=TRUE)  ## K=8 clusters
matchContTable(labels_1, labels_2)
```

---

**NormMixClus**

*Normal mixture model estimation and selection for a series of cluster numbers*

**Description**

Perform co-expression and co-abundance analysis of high-throughput sequencing data, with or without data transformation, using a Normal mixture models. The output of `NormMixClus` is an S4 object of class `RangedSummarizedExperiment`.

**Usage**

```r
NormMixClus(
  y_profiles,
  K,
  subset = NULL,
  parallel = TRUE,
  BPPARAM = bpparam(),
  seed = NULL,
  ...
)
```

**Arguments**

- `y_profiles` *(n x q)* matrix of observed profiles for *n* observations and *q* variables
- `K` Number of clusters (a single value or a sequence of values).
- `subset` Optional vector providing the indices of a subset of genes that should be used for the co-expression analysis (i.e., row indices of the data matrix `y`).
- `parallel` If `FALSE`, no parallelization. If `TRUE`, parallel execution using BiocParallel (see next argument `BPPARAM`). A note on running in parallel using BiocParallel: it may be advantageous to remove large, unneeded objects from the current R environment before calling the function, as it is possible that R’s internal garbage collection will copy these files while running on worker nodes.
- `BPPARAM` Optional parameter object passed internally to `bplapply` when `parallel=TRUE`. If not specified, the parameters last registered with `register` will be used.
seed If desired, an integer defining the seed of the random number generator. If NULL, a random seed is used.

... Additional optional parameters to be passed to NormMixClusK.

Value

An S4 object of class cseqResults, with conditional probabilities of cluster membership for each gene in each model stored as a list of assay data, and corresponding log likelihood, ICL value, number of clusters, and form of Gaussian model for each model stored as metadata.

Author(s)

Andrea Rau, Cathy Maugis-Rabusseau

Examples

```r
## Simulate toy data, n = 300 observations
set.seed(12345)
countmat <- matrix(runif(300*4, min=0, max=500), nrow=300, ncol=4)
countmat <- countmat[which(rowSums(countmat) > 0),]
profiles <- transformRNAseq(countmat, norm="none",
                            transformation="arcsin")$tcounts
conds <- rep(c("A","B","C","D"), each=2)

## Run the Normal mixture model for K = 2,3
## Object of class cseqResults
run <- NormMixClus(y=profiles, K=2:3, iter=5)
run

## Run the Normal mixture model for K=2
## Object of class SummarizedExperiment
run2 <- NormMixClusK(y=profiles, K=2, iter=5)

## Summary of results
summary(run)

## Re-estimate mixture parameters for the model with K=2 clusters
param <- NormMixParam(run, y_profiles=profiles)
```

NormMixClusK Normal mixture model estimation

Description

Perform co-expression and co-abundance analysis of high-throughput sequencing data, with or without data transformation, using a Normal mixture models for single number of clusters K. The output of NormMixClusK is an S4 object of class RangedSummarizedExperiment.
Usage

NormMixClusK(
    y_profiles,
    K,
    alg.type = "EM",
    init.runs = 50,
    init.type = "small-em",
    GaussianModel = "Gaussian_pk_Lk_Ck",
    init.iter = 20,
    iter = 1000,
    cutoff = 0.001,
    verbose = TRUE,
    digits = 3,
    seed = NULL
)

Arguments

y_profiles  y (n x q) matrix of observed profiles for n observations and q variables
K           Number of clusters (a single value).
alg.type    Algorithm to be used for parameter estimation: "EM", "CEM", "SEM"
init.runs   Number of runs to be used for the Small-EM strategy, with a default value of 50
init.type   Type of initialization strategy to be used: "small-em" for the Small-EM strategy, "random", "CEM", or "SEMMax"
GaussianModel One of the 28 forms of Gaussian models defined in Rmixmod, by default equal to the "Gaussian_pk_Lk_Ck" (i.e., a general family model with free proportions, free volume, free shape, and free orientation)
init.iter   Number of iterations to be used within each run for the Small-EM strategy, with a default value of 20
iter        Maximum number of iterations to be run for the chosen algorithm
cutoff      Cutoff to declare algorithm convergence
verbose     If TRUE, verbose output is created
digits      Integer indicating the number of decimal places to be used for the probaPost output
seed        If desired, an integer defining the seed of the random number generator. If NULL, a random seed is used.

Value

An S4 object of class RangedSummarizedExperiment, with conditional probabilities of cluster membership for each gene stored as assay data, and log likelihood, ICL value, number of clusters, and form of Gaussian model stored as metadata.

Author(s)

Cathy Maugis-Rabusseau, Andrea Rau
Examples

```r
## Simulate toy data, n = 300 observations
set.seed(12345)
countmat <- matrix(runif(300*4, min=0, max=500), nrow=300, ncol=4)
countmat <- countmat[which(rowSums(countmat) > 0),]
profiles <- transformRNAseq(countmat, norm="none",
                           transformation="arcsin")$tcounts

conds <- rep(c("A","B","C","D"), each=2)

## Run the Normal mixture model for K = 2,3
## Object of class coseqResults
run <- NormMixClus(y=profiles, K=2:3, iter=5)
run

## Run the Normal mixture model for K=2
## Object of class SummarizedExperiment
run2 <- NormMixClusK(y=profiles, K=2, iter=5)

## Summary of results
summary(run)

## Re-estimate mixture parameters for the model with K=2 clusters
param <- NormMixParam(run, y_profiles=profiles)
```

NormMixParam

**Calculate the mean and covariance for a Normal mixture model**

Description

Calculates the mean and covariance parameters for a normal mixture model of the form $pK_L \mu_k \Sigma_k$

Usage

```r
NormMixParam(
    coseqResults, 
    y_profiles = NULL, 
    K = NULL, 
    digits = 3, 
    plot = FALSE, 
    ... 
)
```

Arguments

- `coseqResults` Object of class coseqResults or RangedSummarizedExperiment (as output from the NormMixClus or NormMixClusK functions)
y_profiles  
\( y \) \((n \times q)\) matrix of observed profiles for \( n \) observations and \( q \) variables, required for \( x \) of class RangedSummarizedExperiment

\( K \)  
The model used for parameter estimation for objects \( x \) of class coseq or NormMixClus. When NULL, the model selected by the ICL criterion is used; otherwise, \( K \) should designate the number of clusters in the desired model

digits  
Integer indicating the number of decimal places to be used for output

plot  
If true, produce heatmaps to visualize the estimated per-cluster correlation matrices

Value

\( \pi \)  
Vector of dimension \( K \) with the estimated cluster proportions from the Gaussian mixture model, where \( K \) is the number of clusters

\( \mu \)  
Matrix of dimension \( K \times d \) containing the estimated mean vector from the Gaussian mixture model, where \( d \) is the number of samples in the data \( y \_\text{profiles} \) and \( K \) is the number of clusters

\( \Sigma \)  
Array of dimension \( d \times d \times K \) containing the estimated covariance matrices from the Gaussian mixture model, where \( d \) is the number of samples in the data \( y \_\text{profiles} \) and \( K \) is the number of clusters

\( \rho \)  
Array of dimension \( d \times d \times K \) containing the estimated correlation matrices from the Gaussian mixture model, where \( d \) is the number of samples in the data \( y \_\text{profiles} \) and \( K \) is the number of clusters

Author(s)

Andrea Rau, Cathy Maugis-Rabusseau

Examples

```r
## Simulate toy data, \( n = 300 \) observations
set.seed(12345)
countmat <- matrix(runif(300*4, min=0, max=500), nrow=300, ncol=4)
countmat <- countmat[which(rowSums(countmat) > 0),]
profiles <- transformRNAseq(countmat, norm="none",
                           transformation="arcsin")$tcounts
conds <- rep(c("A","B","C","D"), each=2)

## Run the Normal mixture model for \( K = 2,3 \)
## Object of class coseqResults
run <- NormMixClus(y=profiles, K=2:3, iter=5)
run

## Run the Normal mixture model for \( K=2 \)
## Object of class SummarizedExperiment0
run2 <- NormMixClusK(y=profiles, K=2, iter=5)

## Summary of results
```
summary(run)

## Re-estimate mixture parameters for the model with K=2 clusters
param <- NormMixParam(run, y_profiles=profiles)

### Description
Plot a coseqResults object.

### Usage

```r
plot(x, ...)
```

```r
# S4 method for signature 'coseqResults'
plot(
  x,
  y_profiles = NULL,
  K = NULL,
  threshold = 0.8,
  conds = NULL,
  average_over_conds = FALSE,
  collapse_reps = "none",
  graphs = c("logLike", "ICL", "profiles", "boxplots", "probapost_boxplots",
             "probapost_barplots", "probapost_histogram"),
  order = FALSE,
  profiles_order = NULL,
  n_row = NULL,
  n_col = NULL,
  add_lines = TRUE,
  ...
)
```

coseqGlobalPlots(object, graphs = c("logLike", "ICL"), ...)

coseqModelPlots(
  probaPost,
  y_profiles,
  K = NULL,
  threshold = 0.8,
  conds = NULL,
  collapse_reps = "none",
  graphs = c("profiles", "boxplots", "probapost_boxplots", "probapost_barplots",
             "probapost_histogram"),
  ...
order = FALSE,
profiles_order = NULL,
n_row = NULL,
n_col = NULL,
add_lines = TRUE,
...)

Arguments

x An object of class "coseqResults"
... Additional optional plotting arguments (e.g., xlab, ylab, use_sample_names, facet_labels)
y_profiles y (n x q) matrix of observed profiles for n observations and q variables to be used for graphing results (optional for logLike, ICL, probapost_boxplots, and probapost_barplots, and by default takes value x$tcounts if NULL)
K If desired, the specific model to use for plotting (or the specific cluster number(s) to use for plotting in the case of coseqModelPlots). If NULL, all clusters will be visualized, and the model chosen by ICL will be plotted
threshold Threshold used for maximum conditional probability; only observations with maximum conditional probability greater than this threshold are visualized
conds Condition labels, if desired
average_over_conds If TRUE, average values of y_profiles within each condition identified by conds for the profiles and boxplots plots. This argument is redundant to collapse_reps = "sum", and collapse_reps should be used instead.
collapse_reps If "none", display all replicates. If "sum", collapse replicates within each condition by summing their profiles. If "average", collapse replicates within each condition by averaging their profiles. For highly unbalanced experimental designs, using "average" will likely provide more easily interpretable plots.
graphs Graphs to be produced, one (or more) of the following: "logLike" (log-likelihood plotted versus number of clusters), "ICL" (ICL plotted versus number of clusters), "profiles" (line plots of profiles in each cluster), "boxplots" (boxplots of profiles in each cluster), "probapost_boxplots" (boxplots of maximum conditional probabilities per cluster), "probapost_barplots" (number of observations with a maximum conditional probability greater than threshold per cluster), "probapost_histogram" (histogram of maximum conditional probabilities over all clusters)
order If TRUE, order clusters in probapost_boxplot by median and probapost_barplot by number of observations with maximum conditional probability greater than threshold
profiles_order If NULL or FALSE, line plots and boxplots of profiles are plotted sequentially by cluster number (K=1, K=2, ...). If TRUE, line plots and boxplots of profiles are plotted in an automatically calculated order (according to the Euclidean distance between cluster means) to plot clusters with similar mean profiles next to one another. Otherwise, the user may provide a vector (of length equal to the number of clusters in the given model) providing the desired order of plots.
n_row  Number of rows for plotting layout of line plots and boxplots of profiles.
n_col  Number of columns for plotting layout of line plots and boxplots of profiles.
add_lines  If TRUE, add red lines representing means to boxplots; if FALSE, these will be suppressed.
object  An object of class "RangedSummarizedExperiment" arising from a call to NormMixClus
probaPost  Matrix or data.frame of dimension \((n \times K)\) containing the conditional probabilities of cluster membership for \(n\) genes in \(K\) clusters arising from a mixture model

Value
Named list of plots of the coseqResults object.

Author(s)
Andrea Rau, Cathy Maugis-Rabusseau

Examples

```r
## Simulate toy data, n = 300 observations
set.seed(12345)
countmat <- matrix(runif(300*4, min=0, max=500), nrow=300, ncol=4)
countmat <- countmat[which(rowSums(countmat) > 0),]
conds <- rep(c("A","B","C","D"), each=2)

## Run the Normal mixture model for K = 2,3,4
run_arcsin <- coseq(object=countmat, K=2:4, iter=5, transformation="arcsin",
                     model="Normal", seed=12345)
run_arcsin

## Plot and summarize results
plot(run_arcsin)
summary(run_arcsin)

## Compare ARI values for all models (no plot generated here)
ARI <- compareARI(run_arcsin, plot=FALSE)

## Compare ICL values for models with arcsin and logit transformations
run_logit <- coseq(object=countmat, K=2:4, iter=5, transformation="logit",
                   model="Normal")
compareICL(list(run_arcsin, run_logit))

## Use accessor functions to explore results
clusters(run_arcsin)
likelihood(run_arcsin)
nbCluster(run_arcsin)
ICL(run_arcsin)

## Examine transformed counts and profiles used for graphing
tcounts(run_arcsin)
profiles(run_arcsin)
```

## Run the K-means algorithm for logclr profiles for K = 2,..., 20
run_kmeans <- coseq(object=countmat, K=2:20, transformation="logclr",
                     model="kmeans")
run_kmeans

---

### Description

A function to summarize the clustering results obtained from a Poisson or Gaussian mixture model estimated using coseq. In particular, the function provides the number of clusters selected for the ICL model selection approach (or alternatively, for the capushe non-asymptotic approach if K-means clustering is used), number of genes assigned to each cluster, and if desired the per-gene cluster means.

### Usage

```r
## S4 method for signature 'coseqResults'
summary(object, y_profiles, digits = 3, ...)
```

### Arguments

- `object` An object of class "coseqResults"
- `y_profiles` Data used for clustering if per-cluster means are desired
- `digits` Integer indicating the number of decimal places to be used for mixture model parameters
- `...` Additional arguments

### Details

Provides the following summary of results:

1) Number of clusters and model selection criterion used, if applicable.
2) Number of observations across all clusters with a maximum conditional probability greater than 90 observations) for the selected model.
3) Number of observations per cluster with a maximum conditional probability greater than 90 (cluster) for the selected model.
4) If desired, the $\mu$ values and $\pi$ values for the selected model in the case of a Gaussian mixture model.

### Value

Summary of the coseqResults object.
Author(s)

Andrea Rau

References


See Also
coseq

Examples

```r
## Simulate toy data, n = 300 observations
set.seed(12345)
countmat <- matrix(runif(300*4, min=0, max=500), nrow=300, ncol=4)
countmat <- countmat[which(rowSums(countmat) > 0),]
conds <- rep(c("A","B","C","D"), each=2)

## Run the Normal mixture model for K = 2,3,4
run_arcsin <- coseq(object=countmat, K=2:4, iter=5, transformation="arcsin", model="Normal", seed=12345)
run_arcsin

## Plot and summarize results
plot(run_arcsin)
summary(run_arcsin)

## Compare ARI values for all models (no plot generated here)
ARI <- compareARI(run_arcsin, plot=FALSE)

## Compare ICL values for models with arcsin and logit transformations
run_logit <- coseq(object=countmat, K=2:4, iter=5, transformation="logit", model="Normal")
compareICL(list(run_arcsin, run_logit))

## Use accessor functions to explore results
clusters(run_arcsin)
likelihood(run_arcsin)
nbCluster(run_arcsin)
ICL(run_arcsin)

tcounts(run_arcsin)
profiles(run_arcsin)

## Examine transformed counts and profiles used for graphing
tcounts(run_arcsin)
profiles(run_arcsin)

## Run the K-means algorithm for logclr profiles for K = 2,..., 20
```
```r
run_kmeans <- coseq(object=countmat, K=2:20, transformation="logclr",
                     model="kmeans")
run_kmeans
```

### transformRNAseq

**Transform RNA-seq data using common transformations**

#### Description

Application of common transformations for RNA-seq data prior to fitting a normal mixture model.

#### Usage

```r
transformRNAseq(
  y,  
  normFactors = "TMM", 
  transformation = "arcsin", 
  geneLength = NA, 
  meanFilterCutoff = NULL, 
  verbose = TRUE 
)
```

#### Arguments

- **y**  
  
  
  *(n x q)* matrix or data.frame of observed counts for *n* observations and *q* variables.

- **normFactors**  
  
  The type of estimator to be used to normalize for differences in library size: “TC” for total count, “DESeq” for the normalization method in the DESeq package, and “TMM” for the TMM normalization method (Robinson and Oshlack, 2010). Can also be a vector (of length *q*) containing pre-estimated library size estimates for each sample.

- **transformation**  
  
  Transformation type to be used: “arcsin”, “logit”, “logMedianRef”, “profile”, “voom”, “logRPKM” (if geneLength is provided by user), “logclr”, “clr”, “alr”, “ilr”, “none”.

- **geneLength**  
  
  Vector of length equal to the number of rows in “y” providing the gene length (bp) for RPKM calculation.

- **meanFilterCutoff**  
  
  Value used to filter low mean normalized counts.

- **verbose**  
  
  If TRUE, include verbose output.

#### Value

- **tcounts**  
  
  Transformed counts.

- **normCounts**  
  
  Normalized counts.

- **snorm**  
  
  Per-sample normalization factors divided by mean normalization factor.

- **ellnorm**  
  
  Per-sample normalization factors.
Examples

```r
set.seed(12345)
countmat <- matrix(runif(300*4, min=0, max=500), nrow=300, ncol=4)
countmat <- countmat[which(rowSums(countmat) > 0),]
conds <- rep(c("A","B","C","D"), each=2)

## Arcsin transformation, TMM normalization
arcsin <- transformRNAseq(countmat, normFactors="TMM", transformation="arcsin")$tcounts

## Logit transformation, TMM normalization
logit <- transformRNAseq(countmat, normFactors="TMM", transformation="logit")$tcounts

## logCLR transformation, TMM normalization
logclr <- transformRNAseq(countmat, normFactors="TMM", transformation="logclr")$tcounts
```
Index

* cluster
coseq-package, 2
* datasets
fietz, 19
* methods
coseq, 10
summary,coseqResults-method, 30
* models
coseq-package, 2
coseqFullResults,coseqResults-method (coseqFullResults), 13
coseqGlobalPlots(plot), 27
coseqModelPlots(plot), 27
coseqResults(coseqResults-class), 16
coseqResults-class, 16
coseqRun, 17,17
DDSEextract(coseqFullResults), 13
DDSEextract,Capushe-method (coseqFullResults), 13
Djumpextract(coseqFullResults), 13
Djumpextract,Capushe-method (coseqFullResults), 13
fietz, 19
ICL(coseqFullResults), 13
ICL,coseqResults-method (coseqFullResults), 13
ICL,MixmodCluster-method (coseqFullResults), 13
ICL,NULL-method (coseqFullResults), 13
ICL,RangedSummarizedExperiment-method (coseqFullResults), 13
kmeansProbaPost, 20
likelihood(coseqFullResults), 13
likelihood,coseqResults-method (coseqFullResults), 13
likelihood,MixmodCluster-method (coseqFullResults), 13
likelihood,NULL-method (coseqFullResults), 13
likelihood,RangedSummarizedExperiment-method (coseqFullResults), 13
logclr, 21
matchContTable, 21
model(coseqFullResults), 13

34
INDEX

model, coseqResults-method
  (coseqFullResults), 13

nbCluster (coseqFullResults), 13
nbCluster, coseqResults-method
  (coseqFullResults), 13
nbCluster, MixmodCluster-method
  (coseqFullResults), 13
nbCluster, NULL-method
  (coseqFullResults), 13
nbCluster, RangedSummarizedExperiment-method
  (coseqFullResults), 13

NormMixClus, 22
NormMixClusK, 23, 23
NormMixParam, 25

plot, 27
plot, coseqResults-method (plot), 27
plot-methods (plot), 27
proba (coseqFullResults), 13
proba, MixmodCluster-method
  (coseqFullResults), 13
profiles (coseqFullResults), 13
profiles, coseqResults-method
  (coseqFullResults), 13

show (coseqFullResults), 13
show, coseqResults-method
  (coseqFullResults), 13
summary (summary, coseqResults-method),
  30
summary, coseqResults-method, 30
summary-methods
  (summary, coseqResults-method),
  30

tcounts (coseqFullResults), 13
tcounts, coseqResults-method
  (coseqFullResults), 13
transformationType (coseqFullResults),
  13
transformationType, coseqResults-method
  (coseqFullResults), 13
transformRNAseq, 32