

# Package ‘TCseq’

April 4, 2025

**Type** Package

**Title** Time course sequencing data analysis

**Version** 1.31.0

**Author** Mengjun Wu <minervajunjun@gmail.com>, Lei Gu <leigu@broadinstitute.org>

**Maintainer** Mengjun Wu <minervajunjun@gmail.com>

**Description** Quantitative and differential analysis of epigenomic and transcriptomic time course sequencing data, clustering analysis and visualization of the temporal patterns of time course data.

**Depends** R (>= 3.4)

**License** GPL (>= 2)

**LazyData** TRUE

**Imports** edgeR, BiocGenerics, reshape2, GenomicRanges, IRanges, SummarizedExperiment, GenomicAlignments, Rsamtools, e1071, cluster, ggplot2, grid, grDevices, stats, utils, methods, locfit

**Suggests** testthat

**biocViews** Epigenetics, TimeCourse, Sequencing, ChIPSeq, RNASeq, DifferentialExpression, Clustering, Visualization

**RoxygenNote** 7.2.3

**git\_url** <https://git.bioconductor.org/packages/TCseq>

**git\_branch** devel

**git\_last\_commit** 2ff417d

**git\_last\_commit\_date** 2024-10-29

**Repository** Bioconductor 3.21

**Date/Publication** 2025-04-03

## Contents

clust-class . . . . .	2
clust.accessors . . . . .	3

countReads . . . . .	4
counts . . . . .	5
countsTable . . . . .	6
DBanalysis . . . . .	6
DBresult . . . . .	8
experiment . . . . .	11
experiment_BAMfile . . . . .	11
genomicIntervals . . . . .	12
peakreference . . . . .	12
TCA-class . . . . .	14
TCA.accessors . . . . .	15
tca_ATAC . . . . .	16
timeclust . . . . .	17
timeclustplot . . . . .	19
timecourseTable . . . . .	20

<b>Index</b>	<b>23</b>
--------------	-----------

---

clust-class	<i>clust class</i>
-------------	--------------------

---

## Description

clust is a S4 class for storing results of the clustering analysis of time course data.

## Details

The clust objects are returned from `timeclust` and have a `show` method printing a compact summary of their contents

## Slots

Object of clust class contains the following slots:

`method` clustering method used

`dist` distance metric used

`data` a matrix of original or standardized data used in the analysis

`centers` a matrix of cluster centers

`cluster` an integer vector of length  $n$  (the integers are the indices of clusters the data points belong to. For the fuzzy cmeans clustering method, a data point is assigned to the closest cluster to which the data point has highest membership value.

`membership` a matrix of membership values of the data points to each clusters

## Author(s)

Mengjun Wu

**See Also**[timeclust](#), @

---

clust.accessors	<i>Accessors to extract slots of a clust class.</i>
-----------------	---

---

**Description**

Accessors are provided to extract data, centers, cluster, and membership slots stored in a `clust` class.

**Usage**

```
clustData(object)

## S4 method for signature 'clust'
clustData(object)

clustCenters(object)

## S4 method for signature 'clust'
clustCenters(object)

clustCluster(object)

## S4 method for signature 'clust'
clustCluster(object)

clustMembership(object)

## S4 method for signature 'clust'
clustMembership(object)
```

**Arguments**

object            clust object object

**Value**

`clustData` returns a data matrix. `clustCenters` returns a matrix of centers. `clustCluster` returns an integer vector. `clustMembership` returns a matrix of membership, see [clust](#) for details.

**Author(s)**

Mengjun Wu

**See Also**[clust](#)


---

countReads	<i>count mapped reads overlap genomic intervals</i>
------------	---

---

**Description**

This function counts mapped reads from multiple BAM files overlapping genomic intervals in genomicFeature in a TCA object. The resulted count table is stored in count slot of the TCA object.

**Usage**

```
countReads(object, dir, method = "summarizeoverlaps", zero.based = TRUE, ...)
```

**Arguments**

object	a TCA object.
dir	character string giving the directory of BAM files.
method	character string giving the counting method. Options are "summarizeOverlaps" and "featureCounts". For Windows system, only "summarizeOverlaps" can be used, For Linux system, both methods can be used.
zero.based	Logical. If TRUE, the start positions of the genomic intervals are <i>0-based</i> , if FALSE, the start positions will be <i>1-based</i> .
...	additional arguments passed to <a href="#">summarizeOverlaps</a> in GenomicAlignments package or <a href="#">featureCounts</a> in Rsubread package.

**Details**

This function provides two options to count the mapped reads: "summarizeOverlaps" in the GenomicAlignments package and "featureCounts" in the Rsubread package. As Rsubread package is only available for linux systems, Windows users can only choose "summarizeOverlaps". The user could further customize the counting paramters by passing additional arguments (...), otherwise the default settings of the two methods will be used. For details of the counting parameters, see [summarizeOverlaps](#), [featureCounts](#).

**Value**

A TCA object with updated count slot.

**Author(s)**

Mengjun Wu

**See Also**

[summarizeOverlaps](#), [featureCounts](#)

---

counts *Extracts counts of a TCA object.*

---

### Description

counts extract raw read counts stored in a TCA object or compute normalized counts from the raw counts.

### Usage

```
## S4 method for signature 'TCA'  
counts(object, normalization = "none", lib.norm = TRUE, log = FALSE, ...)
```

```
## S4 replacement method for signature 'TCA'  
counts(object) <- value
```

### Arguments

object	a TCA object.
normalization	character string giving the normalization method. Options are "none" (original raw counts), "cpm" (counts per million), "rpkm" (reads per kilobase per million).
lib.norm	logical indicating whether or not use effective library size (see Details below) when normalization is "cpm" or "rpkm".
log	logical if TRUE, the returned value will be on a log <sub>2</sub> scale.
...	additional arguments passed to <a href="#">cpm</a> or <a href="#">rpkm</a> in the edgeR package.
value	an integer matrix.

### Details

when calculating normalized counts, library size can be rescaled to minimize the log-fold changes between samples for most genomic features (e.g. genes, binding sites) by multiplying a scale factor. The rescaled library size is called effective library size. In this function, the scale factor is calculated using the weighted trimmed mean of M-values (TMM, Robinson et al (2010))

If log<sub>2</sub> values are computed, a small count would be added to avoid logarithm of zero. The actual added count will be scaled according to the library size, for details see [addPriorCount](#) in the edgeR package when not specified, the prior count is set to 0.25 by default.

### Value

An integer matrix

### Author(s)

Mengjun Wu

**References**

Robinson, M. D., & Oshlack, A. (2010). A scaling normalization method for differential expression analysis of RNA-seq data. *Genome biology*, 11(3), 1.

**Examples**

```
data(tca_ATAC)
c <- counts(tca_ATAC)
# normalized counts table
c_norm <- counts(tca_ATAC, normalization='rpkm')
```

---

countsTable	<i>An example read Counts table</i>
-------------	-------------------------------------

---

**Description**

A dataset of exemplary read counts

**Usage**

```
data(countsTable)
```

**Format**

A data frame containing experiment design information for 12 samples/libraries.

**Value**

A data frame

**Examples**

```
data(countsTable)
```

---

DBanalysis	<i>Perform differential expression analysis</i>
------------	---

---

**Description**

This function is a wrapper for the [glmFit](#) in edgeR package.

**Usage**

```
DBanalysis(
  object,
  categories = "timepoint",
  norm.lib = TRUE,
  filter.type = NULL,
  filter.value = NULL,
  samplePassfilter = 2,
  ...
)
```

**Arguments**

<code>object</code>	a TCA object.
<code>categories</code>	character string giving which column in design will be used for differential analysis. For time course analysis, the default column is "timepoint".
<code>norm.lib</code>	logical indicating whether or not use effective library size when perform normalization. See <a href="#">counts</a> for more details.
<code>filter.type</code>	character string indicating which type of count (raw or normalized) is used when performing filtering. Options are "raw", "cpm", "rpkm", "NULL". No filtering will be performed when using "NULL".
<code>filter.value</code>	a numeric value; minimum values of selected <code>filter.type</code> ("raw", "cpm", "rpkm"). It is used in combination with <code>samplePassfilter</code> .
<code>samplePassfilter</code>	a numeric value indicating the minimum number of samples/libraries in which a genomic feature has counts value (raw or normalized) more than <code>filter.value</code> . Smaller than this number, the genomic feature will be filtered out.
<code>...</code>	additional arguments passed to <a href="#">glmFit</a> from edgeR package.

**Details**

The differential event is detected by using the generalized linear model (GLM) methods (McCarthy et al, 2012). This function fits the read counts of each genes to a negative binomial glm by using [glmFit](#) function from edgeR. To further test the significance of changes, see [DBresult](#), [TopDBresult](#)

**Value**

A TCA object

**Author(s)**

Mengjun Wu, Lei Gu

**References**

McCarthy,D.J.,Chen, Y., & Smyth, G. K.(2012). Differential expression analysis of multifactor RNA-Seq experiments with respect to biological variation. *Nucleic acids research* 40, 4288-4297.

**See Also**

DBresult, TopDBresult

**Examples**

```
data(tca_ATAC)
tca_ATAC <- DBanalysis(tca_ATAC)
```

---

DBresult

*This function tests for differential expression*

---

**Description**

This function is a wrapper for [glmLRT](#) in edgeR package. It performs likelihood ratio tests for given coefficients contrasts after fitting read counts to a negative binomial glm by [DBanalysis](#). DBresult also extracts the differential analysis results of given contrasts at a chosen significance level. DBresult.cluster returns similar results but only contain genomic features belong to a given cluster.

**Usage**

```
DBresult(  
  object,  
  group1 = NULL,  
  group2 = NULL,  
  contrasts = NULL,  
  p.adjust = "fdr",  
  top.sig = FALSE,  
  pvalue = "paj",  
  pvalue.threshold = 0.05,  
  abs.fold = 2,  
  direction = "both",  
  result.type = "GRangesList"  
)
```

```
DBresult.cluster(  
  object,  
  group1 = NULL,  
  group2 = NULL,  
  contrasts = NULL,  
  p.adjust = "fdr",  
  top.sig = FALSE,  
  pvalue = "paj",  
  pvalue.threshold = 0.05,  
  abs.fold = 2,  
  direction = "both",
```



```

    cluster,
    cmthreshold = NULL,
    result.type = "GRangesList"
)

```

### Arguments

<code>object</code>	a TCA object, for <code>DBresult</code> , <code>DBanalysis</code> should already be called on the object; for <code>DBresult.cluster</code> , both <code>DBanalysis</code> and <code>timeclust</code> should be already called.
<code>group1</code>	character string giving the group to be compared with, i.e., the denominator in the fold changes. <code>group1</code> can be set <code>NULL</code> and will be ignored if the comparisons are passed to <code>contrasts</code>
<code>group2</code>	a character vector giving the other groups to compare with <code>group1</code> , i.e., the numerator in the fold changes. <code>group2</code> can be set <code>NULL</code> and will be ignored if the comparisons are passed to <code>contrasts</code>
<code>contrasts</code>	a character vector, each string in the vector gives a contrast of two groups with the format "group2vsgroup1", <code>group1</code> is the denominator level in the fold changes and <code>group2</code> is the numerator level in the fold changes.
<code>p.adjust</code>	character string specifying a correction method for p-values. Options are "holm", "hochberg", "hommel", "bonferroni", "BH", "BY", "fdr", and "none".
<code>top.sig</code>	logical if <code>TRUE</code> , only genomic regions with given log2-fold changes and significance levels (p-value) will be returned. Log2-fold changes are defined by <code>abs.fold</code> and <code>direction</code> ; significance levels are defined by <code>pvalue</code> and <code>pvalue.threshold</code>
<code>pvalue</code>	character string specify the type of p-values used for defining the significance level ( <code>PValue</code> or adjusted p-value <code>paj</code> )
<code>pvalue.threshold</code>	a numeric value giving threshold of selected p-value, Significant changes have lower (adjusted) p-values than the threshold.
<code>abs.fold</code>	a numeric value, the minimum absolute log2-fold changes. The returned genomic regions have changes with absolute log2-fold changes exceeding <code>abs.fold</code> .
<code>direction</code>	character string specify the direction of fold changes. "up": positive fold changes; "down": negative fold changes; "both": both positive and negative fold changes.
<code>result.type</code>	character string giving the data type of return value. Options are "GRangesList" and "list".
<code>cluster</code>	an integer giving the number of cluster from which genomic features are extracted.
<code>cmthreshold</code>	a numeric value, this argument is applicable only if <code>cmeans</code> ' clustering method is selected when calling <code>timeclust</code> function. if not <code>NULL</code> , the result table of genomic features that belong to the defined <code>cluster</code> and the membership values to this cluster exceed <code>cmthreshold</code> are extracted.

### Details

This function uses [glmLRT](#) from `edgeR` which perform likelihood ratio tests for the significance of changes. For more details, see [glmLRT](#)

**Value**

A list or a GRangesList. If `result.type` is "GRangesList", a GRangesList is returned containing the differential analysis results for all provided contrasts. Each GRanges object of the list is one contrast, the analysis results are contained in 4 metadata columns:

`logFC` log2-fold changes between two groups.

`PValue` p-values.

`paaj` adjusted p-values

`id` name of genomic features

If `result.type` is "list", a list of data frames is returned. Each data frame contains one contrast with the following columns:

`logFC` log2-fold changes between two groups.

`PValue` p-values.

`paaj` adjusted p-values

`chr` name of chromosomes

`start` starting positions of features in the chromosomes

`end` ending positions of features in the chromosomes

`id` name of genomic features

**Note**

If not NULL `group1`, `group2` and `contrasts`, result tables are extracted from comparisons in `contrasts`.

**Author(s)**

Mengjun Wu, Lei Gu

**See Also**

[glmLRT](#)

**Examples**

```
data(tca_ATAC)
tca_ATAC <- DBanalysis(tca_ATAC)
### extract differential analysis of 24h, 72h to 0h
# set the contrasts using the 'group1' and 'group2' parameters
res1 <- DBresult(tca_ATAC, group1 = '0h', group2 = c('24h', '72h'))
# one can get the same result by setting the contrasts using the 'contrasts' parameter
res2 <- DBresult(tca_ATAC, contrasts = c('24hvs0h', '72hvs0h'))
# extract significant differential events
res.sig <- DBresult(tca_ATAC, contrasts = c('24hvs0h', '72hvs0h'),
                  top.sig = TRUE)

# extract differential analysis of 24h, 72h to 0h of a given cluster
tca_ATAC <- timecourseTable(tca_ATAC, filter = TRUE)
```

```
tca_ATAC <- timeclust(tca_ATAC, algo = 'cm', k = 6)
res_cluster1 <- DBresult.cluster(tca_ATAC, group1 = '0h',
                                group2 = c('24h', '72h'),
                                cluster = 1)
```

---

experiment

*An example experiment design without BAM file information*

---

### **Description**

A dataset of exemplary experiment design without BAM file information

### **Usage**

```
data(experiment)
```

### **Format**

A data frame containing experiment design information for 12 samples/libraries.

### **Value**

A data frame

### **Examples**

```
data(experiment)
```

---

experiment\_BAMfile

*An example experiment design with BAM file information*

---

### **Description**

A dataset of exemplary experiment design with BAM file information

### **Usage**

```
data(experiment_BAMfile)
```

### **Format**

A data frame containing experiment design information for 12 samples/libraries.

**Value**

A data frame

**Examples**

```
data(experiment_BAMfile)
```

---

genomicIntervals	<i>An example reference genomic regions</i>
------------------	---

---

**Description**

A dataset of exemplary genomic regions

**Usage**

```
data(genomicIntervals)
```

**Format**

A data frame containing 2751 genomic regions.

**Value**

A data frame

**Examples**

```
data(genomicIntervals)
```

---

peakreference	<i>combine and merge multiple BED files</i>
---------------	---

---

**Description**

This function merges overlapping genomic regions into a single feature. The merged single feature represents the widest genomic interval that covers all overlapping regions.

**Usage**

```
peakreference(  
  data = NULL,  
  dir = NULL,  
  pattern = NULL,  
  merge = TRUE,  
  overlap = 1,  
  ratio = NULL  
)
```

**Arguments**

data	a data frame containing coordinates information of peaks to be merged. Columns of the data frame should be consistent with the BED format where the first column contains chromosome information, the second column the starting position, and the third column the ending position.
dir	a character string giving the directory where BED files are stored. If data is not given, the function will read in the BED files under code.
pattern	an <a href="#">regular expression</a> , only files that have names match the regular expression will be read in.
merge	logical indicating whether to merge overlapped regions or not. If False, regions are simply combined.
overlap	a numeric value giving the least number of base(s) two regions should overlap when merging them.
ratio	a numeric value giving the threshold of overlapping ratio between two regions to merge them. See 'Details' below for the definition of the overlapping ratio.

**Details**

The overlapping ratio (OR) is defined as:

$$OR = \frac{n}{\min(\text{length}(a), \text{length}(b))}$$

$a, b$  are two genomic regions,  $n$  is the number of overlapping bases between region  $a$  and region  $b$ .

**Value**

a data frame with four columns: chr, start, stop, id

**Author(s)**

Mengjun Wu, Lei Gu

**Examples**

```
peaks <- data.frame(chr = c(rep('chr1', 4), rep('chr2', 3), rep('chr3', 2)),
  start = c(100, 148, 230, 300, 330, 480, 1000, 700, 801),
  end = c(150, 220, 500, 450, 600, 900, 1050, 760, 900))

merged_peaks <- peakreference(data = peaks, merge = TRUE, overlap = 1)
```

TCA-class

*TCA class and constructor***Description**

TCA is a S4 class for storing input data, results of differential analysis and clustering analysis. A TCA object can be created by the constructor function taking a table of sample information, a table of the genomic coordinates of features, and read count table (optional).

**Usage**

```
TCA(design, counts = matrix(0L, 0L, 0L), genomicFeature, zero.based = TRUE)
```

```
TCAFromSummarizedExperiment(se, genomicFeature = NULL)
```

**Arguments**

design	a data frame containing information of samples/libraries. For time course analysis, design table should contain at least three columns (case insensitive): <code>sampleid</code> , <code>timepoint</code> and <code>group</code> providing time point and group information of each sample/library. If <code>counts</code> is not provided when creating TCA object, an optional column <code>bamfile</code> can be used to provide BAM filename of each sample/library and generate count table using <code>countReads</code> function later.
counts	an integer matrix containing read counts. Rows correspond to genomic features and columns to samples/libraries. The name of column <code>s</code> should be the same as the time points in design.
genomicFeature	a data frame or a <code>GRanges</code> object containing genomic coordinates of features of interest (e.g. genes in RNA-seq, binding regions in ChIP-seq). If <code>genomicFeature</code> is a data frame, four columns are required in <code>genomicFeature</code> : <code>id</code> , <code>chr</code> , <code>start</code> , <code>end</code> ; if <code>genomicFeature</code> is a <code>GRanges</code> object, the metadata column "id" is required. For <code>TCAFromSummarizedExperiment</code> , <code>genomicFeature</code> must be provided if <code>se</code> is a <code>SummarizedExperiment</code> object.
zero.based	Logical. If <code>TRUE</code> , the start positions of the genomic ranges in the returned TCA object are <i>0-based</i> , if <code>FALSE</code> , the start positions will be <i>1-based</i> .
se	A <code>SummarizedExperiment</code> or a <code>RangedSummarizedExperiment</code> object. The object might contain multiple assays in the assay list, only the first one will be taken to construct TCA object. For <code>SummarizedExperiment</code> object, <code>genomicFeature</code> must be provided while for <code>RangedSummarizedExperiment</code> object, the genomic features will be extracted directly from the object.

**Details**

A TCA object can be created without providing read counts, read counts can be provided by `counts` or generated by `countReads`. For the read counts, the number of rows should equal to that in `genomicFeature` and the number of columns should equal to number of rows in `design`; in addition, the name of column names should be the same as the time points in design. Input data

and analysis results in a TCA object can be accessed by using corresponding accessors and functions. The TCA objects also have a show method printing a compact summary of their contents see [counts](#), [TCA.accessors](#), [DBresult](#), [tcTable](#), [timeclust](#), [clust](#)

### Value

A TCA object

### Author(s)

Mengjun Wu

### See Also

[counts](#), [TCA.accessors](#), [DBresult](#), [timeclust](#), [clust](#)

### Examples

```
#create data frame of experiment design: 4 time points and 2 replicates for each time point.
d <- data.frame(sampleID = 1:8, group = rep(c(1, 2, 3, 4), 2),
               timepoint = rep(c('0h', '24h', '48h', '72h'), 2))

#create data frame of genomic intervals of interest
gf <- data.frame(chr = c(rep('chr1', 3), rep('chr2', 2), rep('chr4', 2)),
               start = seq(100, 2000, by = 300),
               end = seq(100, 2000, by = 300) + 150,
               id = paste0('peak', 1:7))
tca <- TCA(design = d, genomicFeature = gf)
genomicFeature(tca)

#if count table is available
c <- matrix(sample(1000, 56), nrow = 7, dimnames = list(paste0('peak', 1:7), 1:8))
tca <- TCA(design = d, counts = c, genomicFeature = gf)
# replace the count table of a \code{TCA} object
c2 <- matrix(sample(500, 56), nrow = 7, dimnames = list(paste0('peak', 1:7), 1:8))
counts(tca) <- c2
```

---

TCA.accessors

*Accessors to extract slots of a TCA class.*

---

### Description

Accessors are provided to extract design, genomicFeature, tcTable, clustResults slots of a TCA class. The design slot stores experimental information of samples/libraries, the genomicFeature slot stores genomic coordinates of features, the tcTable slot stores time course data as a matrix, where rows are genomic features and columns time points. The clustResults slot stores results of clustering analysis as a clust object.

**Usage**

```
## S4 method for signature 'TCA'  
design(object)  
  
genomicFeature(object)  
  
tcTable(object)  
  
## S4 method for signature 'TCA'  
tcTable(object)  
  
clustResults(object)  
  
## S4 method for signature 'TCA'  
clustResults(object)
```

**Arguments**

object            TCA object object

**Value**

design returns a data frame. genomicFeature returns a data frame. tcTable returns a numeric matrix. clustResults returns a clust object, see [clust](#) for details.

**Author(s)**

Mengjun Wu

**See Also**

[clust](#)

**Examples**

```
data(tca_ATAC)  
genomicFeature(tca_ATAC)  
tcTable(tca_ATAC)
```

---

tca\_ATAC

*An example TCA object*

---

**Description**

A TCA object storing exemplary ATAC-seq time course data, including the experiment design, read counts, reference genomic regions.



**Usage**

```
data(tca_ATAC)
```

**Format**

A TCA object of exemplary ATAC-seq time course data

**Value**

A TCA object

**Examples**

```
data(tca_ATAC)
```

---

timeclust	<i>time course data clustering</i>
-----------	------------------------------------

---

**Description**

This function performs clustering analysis of the time course data.

**Usage**

```
timeclust(  
  x,  
  algo,  
  k,  
  dist = "distance",  
  dist.method = "euclidean",  
  centers = NULL,  
  standardize = TRUE,  
  ...  
)
```

**Arguments**

x	a TCA object returned from <code>timecourseTable</code> or a matrix
algo	a character string giving a clustering method. Options are "km" (kmeans), "pam" (partitioning around medoids), "hc" (hierachical clustering), "cm" (cmeans).
k	a numeric value between 1 and $n - 1$ ( $n$ is the number of data points to be clustered).
dist	a character string specifying either "distance" or "correlation" will be used to measure the distance between data points.

<code>dist.method</code>	a character string. It can be chosen from one of the correlation methods in <code>cor</code> function ("pearson", "spearman", "kendall") if <code>dist</code> is "correlation", or one of the distance measure methods in <code>dist</code> function (for example, "euclidean", "manhattan") if <code>dist</code> is "distance".
<code>centers</code>	a numeric matrix giving initial centers for <code>kmeans</code> , <code>pam</code> or <code>cmeans</code> . If given, number of rows of the matrix must be equal to <code>k</code> .
<code>standardize</code>	logical, if TRUE, z-score transformation will be performed on the data before clustering. See 'Details' below.
<code>...</code>	additional arguments passing to <code>kmeans</code> , <code>pam</code> , <code>hclust</code> , <code>cmeans</code>

### Details

two types of clustering methods are provided: hard clustering (`kmeans`, `pam`, `hclust`) and soft clustering (`cmeans`). In hard clustering, a data point can only be allocated to exactly one cluster (for `hclust`, `cutree` is used to cut a tree into clusters), while in soft clustering (also known as fuzzy clustering), a data point can be assigned to multiple clusters, membership values are used to indicate to what degree a data point belongs to each cluster.

To better capture the differences of temporal patterns rather than expression levels, z-score transformation can be applied to convert the expression values to z-scores by performing the following formula:

$$z = \frac{x - \mu}{\sigma}$$

$x$  is the value to be converted (e.g., expression value of a genomic feature in one condition),  $\mu$  is the population mean (e.g., average expression value of a genomic feature across different conditions),  $\sigma$  is the standard deviation (e.g., standard deviation of the expression values of a genomic feature across different conditions).

### Value

If `x` is a TCA object, a TCA object will be returned. If `x` is a matrix, a `cLust` object will be returned

### Author(s)

Mengjun Wu

### See Also

`clust`, `kmeans`, `pam`, `hclust`, `cutree`

### Examples

```
example.mat <- matrix(rnorm(1600,sd=0.3), nrow = 200,
                     dimnames = list(paste0('peak', 1:200), 1:8))
clust_res <- timeclust(x = example.mat, algo = 'cm', k = 4)
# return a cLust object
```

---

timeclustplot	<i>Plot clustering results for time course data.</i>
---------------	--

---

## Description

This function plots the clusters generated from `timeclust`. For fuzzy cmeans clustering, data points are color-coded according to membership values, the color palettes can be customized.

## Usage

```
timeclustplot(  
  object = NULL,  
  categories = "timepoint",  
  value = "expression",  
  cols = NULL,  
  cl.color = "gray50",  
  membership.color = rainbow(30, s = 3/4, v = 1, start = 1/6),  
  title.size = 18,  
  axis.line.size = 0.6,  
  axis.title.size = 18,  
  axis.text.size = 16,  
  legend.title.size = 14,  
  legend.text.size = 14  
)
```

## Arguments

<code>object</code>	a TCA object or a <code>clust</code> object
<code>categories</code>	character string giving the x-axis label
<code>value</code>	character string giving the y-axis label
<code>cols</code>	integer value specifying number of columns in the final layout.
<code>cl.color</code>	character string specifying a color for hard clustering.
<code>membership.color</code>	color palettes, a character vector of n colors
<code>title.size</code>	numeric value specifying the font size of title of each plot in the layout
<code>axis.line.size</code>	numeric value specifying the size of both axis lines
<code>axis.title.size</code>	numeric value specifying the font size of titles of both axis
<code>axis.text.size</code>	numeric value specifying the font size of labels of both axis
<code>legend.title.size</code>	numeric value specifying the font size of legend title
<code>legend.text.size</code>	numeric value specifying the font size of legend text

**Value**

Plot all clusters in one plot and return a list of ggplot objects, each object is for one cluster. The ggplot object can be drawn by calling `print.ggplot`

**Author(s)**

Mengjun Wu

**Examples**

```
x <- matrix(sample(500, 1600, replace = TRUE), nrow = 200,
            dimnames = list(paste0('peak', 1:200), 1:8))
clust_res <- timeclust(x, algo = 'cm', k = 4, standardize = TRUE)
p <- timeclustplot(clust_res, cols = 2)
# to plot a individual cluster
print (p[[2]]) # plot cluster 2
print (p[[3]]) # plot cluster 3
```

---

timecourseTable

*constructs time course table for clustering analysis*

---

**Description**

This function constructs a time course table of which rows are genomic features and columns time points. values can be normalized expression levels or log<sub>2</sub>-fold changes compared to a control time point. The time course table is used for clustering analysis.

**Usage**

```
timecourseTable(
  object,
  value = "expression",
  control.group = NULL,
  lib.norm = TRUE,
  norm.method = "rpkm",
  subset = NULL,
  filter = FALSE,
  pvalue = "fdr",
  pvalue.threshold = 0.05,
  abs.fold = 2,
  direction = "both",
  ...
)
```

**Arguments**

<code>object</code>	a TCA object returned by <code>DBanalysis</code> .
<code>value</code>	a character string, either "expression" or "FC". "expression" is the mean normalized read counts of replicates, "FC" is the log <sub>2</sub> -fold changes compared to the first time point.
<code>control.group</code>	a character string giving the time point to be compared with, i.e., the denominator in the fold changes. It should match one of the time points in the design table in the TCA object.
<code>lib.norm</code>	logical indicating whether or not use effective library size (see "Details" in <a href="#">counts</a> ).
<code>norm.method</code>	a character string specifying the normalization method if <code>value</code> is "expression"
<code>subset</code>	an optional character vector giving a subset of genomic features, if not NULL, time course table is generated for only this subset of genomic features.
<code>filter</code>	logical, whether to drop the genomic features shows no significant changes (defined by <code>pvalue</code> , <code>pvalue.threshold</code> , <code>abs.fold</code> and <code>direction</code> ) between any two time points.
<code>pvalue</code>	character string specify the type of p-values: "none" is unadjusted p-value or one of adjusted p-value "holm", "hochberg", "hommel", "bonferroni", "BH", "BY", "fdr".
<code>pvalue.threshold</code>	a numeric value giving threshold of selected p-value, significant changes have lower (adjusted) p-values than the threshold.
<code>abs.fold</code>	a numeric value, the least minimum log <sub>2</sub> -fold changes. The returned genomic regions have changes with absolute log <sub>2</sub> -fold changes exceeding <code>abs.fold</code> .
<code>direction</code>	character string specify the direction of fold changes. "up": positive fold changes; "down": negative fold changes; "both": both positive and negative fold changes.
<code>...</code>	additional arguments passing to <a href="#">rpkm</a> , <a href="#">cpm</a>

**Value**

A TCA object

**Note**

If "expression" in `value` is chosen, the average normalized expression values of replicates for each group will be calculated and returned.

**Author(s)**

Mengjun Wu



# Index

## \* datasets

- countsTable, 6
  - experiment, 11
  - experiment\_BAMfile, 11
  - genomicIntervals, 12
  - tca\_ATAC, 16
- addPriorCount, 5
- clust, 3, 4, 15, 16, 18
- clust (clust-class), 2
- clust-class, 2
- clust.accessors, 3
- clustCenters (clust.accessors), 3
- clustCenters, clust-method  
(clust.accessors), 3
- clustCluster (clust.accessors), 3
- clustCluster, clust-method  
(clust.accessors), 3
- clustData (clust.accessors), 3
- clustData, clust-method  
(clust.accessors), 3
- clustMembership (clust.accessors), 3
- clustMembership, clust-method  
(clust.accessors), 3
- clustResults (TCA.accessors), 15
- clustResults, TCA-method  
(TCA.accessors), 15
- cmeans, 18
- cor, 18
- countReads, 4, 14
- counts, 5, 7, 14, 15, 21
- counts, TCA-method (counts), 5
- counts<-, TCA-method (counts), 5
- countsTable, 6
- cpm, 5, 21
- cutree, 18
- DBanalysis, 6, 8
- DBresult, 8, 15
- design (TCA.accessors), 15
- design, TCA-method (TCA.accessors), 15
- dist, 18
- experiment, 11
- experiment\_BAMfile, 11
- featureCounts, 4
- genomicFeature (TCA.accessors), 15
- genomicFeature, TCA-method  
(TCA.accessors), 15
- genomicIntervals, 12
- glmFit, 6, 7
- glmLRT, 8–10
- hclust, 18
- kmeans, 18
- pam, 18
- peakreference, 12
- print.ggplot, 20
- rpkm, 5, 21
- summarizeOverlaps, 4
- TCA (TCA-class), 14
- TCA-class, 14
- TCA.accessors, 15, 15
- tca\_ATAC, 16
- TCAFromSummarizedExperiment  
(TCA-class), 14
- tcTable, 15
- tcTable (TCA.accessors), 15
- tcTable, TCA-method (TCA.accessors), 15
- timeclust, 2, 3, 9, 15, 17, 19
- timeclustplot, 19
- timecourseTable, 17, 20