Package: scCompoundDE (via r-universe)

July 9, 2026

Type Package

Title Compositional and Transcriptional Decomposition of Pseudo-Bulk Differential Expression

Version 0.99.1

Description scCompoundDE decomposes pseudo-bulk differential expression (DE) signals into two orthogonal components: transcriptional changes (cell-intrinsic expression shifts) and compositional changes (shifts in the relative abundance of cell subtypes). Standard pseudo-bulk DE tools confound these two sources of signal, producing spurious DE calls when subtype proportions differ between conditions. scCompoundDE fits per-subtype limma-voom models, estimates subtype proportion shifts, and uses a z-score-normalized decomposition to assign each gene a TC_ratio score — the fraction of its DE signal attributable to transcription versus composition. Genes are then classified as transcriptional (real biology), compositional (artifact), or mixed (requires caution). All functions operate natively on SingleCellExperiment objects and return a CDEResult S4 object that extends the standard DE output with full decomposition statistics.

License MIT + file LICENSE

URL https://github.com/SubhadipJana1409/scCompoundDE

BugReports https://github.com/SubhadipJana1409/scCompoundDE/issues

biocViews SingleCell, DifferentialExpression, StatisticalMethod, GeneExpression, Transcriptomics, CellBasedAssays, Sequencing, WorkflowStep, Transcription

Encoding UTF-8

Depends R (>= 4.5.0)

Imports SingleCellExperiment, SummarizedExperiment, S4Vectors, BiocParallel, Matrix, limma, methods, stats, ggplot2, rlang, utils

Suggests BiocStyle, knitr, rmarkdown, testthat (>= 3.0.0), scuttle, withr

VignetteBuilder knitr

Config/testthat/edition 3

Config/roxygen2/version 8.0.0

Config/pak/sysreqs zlib1g-dev

Repository https://bioc.r-universe.dev

Date/Publication 2026-06-30 10:15:57 UTC

RemoteUrl https://github.com/bioc/scCompoundDE

RemoteRef HEAD

RemoteSha baa70eacf60c62657a90813dc621c0f63a1449c6

Contents

`scCompoundDE-package` *scCompoundDE: Compositional and Transcriptional Decomposition of Pseudo-Bulk Differential Expression*

Description

scCompoundDE decomposes pseudo-bulk differential expression (DE) signals into two orthogonal components: **transcriptional** changes (cell-intrinsic expression shifts) and **compositional** changes (shifts in the relative abundance of cell subtypes within a broad population).

Standard pseudo-bulk DE tools treat a pseudo-bulk sample as if it were homogeneous, confounding true transcriptional change with artifactual signal arising from subtype proportion shifts. For example, if T cells in disease donors are predominantly exhausted (high PDCD1, TOX) while T cells in healthy donors are predominantly naive (high IL7R, CCR7), a standard pseudo-bulk DE analysis will report PDCD1 and IL7R as significantly DE – but these genes changed because the *composition* of T cells changed, not because T cells themselves changed their transcriptome. scCompoundDE detects and quantifies this confound for every tested gene.

Key innovation

For each gene, scCompoundDE computes a **TC_ratio** score in `[0, 1]`:

- **TC_ratio ~= 1** – gene is driven by cell-intrinsic transcriptional change (real biology).
- **TC_ratio ~= 0** – gene is driven by a shift in subtype proportions (compositional artefact).
- **TC_ratio ~= 0.5** – mixed signal (both components contribute).

Main functions

`compoundDE` Run the full decomposition pipeline. Returns a `CDEResult` S4 object.

`filterGenesBySource` Extract transcriptional, compositional, or mixed gene lists from a `CDEResult`.

`plotDecomposition` Scatter plot of T_score vs C_score for all genes.

`plotProportion` Stacked bar chart of subtype proportions per condition.

`plotTCRatio` Histogram of TC_ratio distribution with classification thresholds.

How it fits into a scRNA-seq workflow

- Use `scBatchQC` first to flag and remove low-quality cells.
- Use `scFastDE` or any pseudo-bulk tool to get DE genes.
- Use `scCompoundDE` to validate whether those DE genes are transcriptionally driven or compositional artefacts.

Author(s)

Maintainer: Subhadip Jana <subhadipjana1409@gmail.com> (ORCID)

Authors:

- Subhadip Jana <subhadipjana1409@gmail.com> (ORCID)

References

Crowell HL et al. (2020). muscat detects subpopulation-specific state transitions from multi-sample multi-condition single-cell transcriptomics data. *Nature Communications*, 11, 6077.

Then E et al. (2023). Distinguishing cell type composition and cell type-specific effects in bulk tissues. *bioRxiv*.

See Also

Useful links:

- https://github.com/SubhadipJana1409/scCompoundDE
- Report bugs at https://github.com/SubhadipJana1409/scCompoundDE/issues

CDEResult *Constructor for CDEResult*

Description

Create a new CDEResult object.

Usage

```
CDEResult(deTable, subtypeProportions, subtypeDE, params = list())
```

Arguments

deTable	A DataFrame of per-gene compound DE statistics.
subtypeProportions	A matrix of subtype proportions (samples x subtypes).
subtypeDE	A named list of per-subtype DE DataFrames.
params	A list of analysis parameters.

Value

A CDEResult object.

Examples

```
library(S4Vectors)
dt <- DataFrame(
    gene      = c("G1", "G2"),
    logFC     = c(1.2, 0.1),
    AveExpr   = c(3.1, 2.0),
    t         = c(4.1, 0.5),
    P.Value   = c(0.001, 0.8),
    adj.P.Val = c(0.01, 0.9),
    B         = c(2.1, -2.0),
    T_score   = c(1.1, 0.05),
    C_score   = c(0.1, 0.09),
    T_score_z = c(1.5, 0.2),
    C_score_z = c(0.2, 0.3),
    TC_ratio  = c(0.88, 0.40),
    source    = c("transcriptional", "mixed")
)
pm <- matrix(c(0.6, 0.4, 0.3, 0.7), nrow = 2,
             dimnames = list(c("D1___ctrl","D1___treat"),
                             c("TypeA","TypeB")))
obj <- CDEResult(deTable = dt, subtypeProportions = pm,
                 subtypeDE = list(), params = list(broad_type = "T_cell"))
obj
```

CDEResult-class *CDEResult: Compound DE Result Container*

Description

An S4 class storing the output of `compoundDE`. Every pseudo-bulk DE gene is assigned a `TC_ratio` score – the fraction of its observed fold-change attributable to transcriptional (cell-intrinsic) versus compositional (subtype proportion shift) signal.

Slots

`deTable` A `DataFrame` with one row per gene containing `logFC`, `AveExpr`, `t`, `P.Value`, `adj.P.Val`, `B` (from the broad limma model), `T_score`, `C_score`, `T_score_z`, `C_score_z`, `TC_ratio`, and `source` (transcriptional / compositional / mixed).

`subtypeProportions` A `matrix` of subtype proportions with rows = samples (donor___condition) and columns = subtypes.

`subtypeDE` A named `list` of per-subtype `DataFrames` from the limma DE models.

`params` A `list` of analysis parameters.

compoundDE *Compound Differential Expression: Transcriptional and Compositional Decomposition*

Description

`compoundDE` decomposes pseudo-bulk differential expression (DE) signals into two orthogonal components:

- **Transcriptional (T):** Cell-intrinsic expression changes – the gene would be DE even if subtype proportions were held fixed.
- **Compositional (C):** Signal arising from a shift in the relative abundance of subtypes – the gene appears DE only because high-expressing (or low-expressing) subtypes became more or less common.

Standard pseudo-bulk DE tools (DESeq2, edgeR, limma-voom) confound these two sources. `compoundDE` fits a separate limma-voom model for each cell subtype within the broad population, estimates subtype proportion changes across conditions, and uses a z-score-normalised decomposition to assign each gene a **TC_ratio** – the fraction of its DE signal attributable to transcriptional change.

Usage

```
compoundDE(
  sce,
  broad_type,
  subtype_col,
  broad_col,
  donor,
  condition,
  contrast = NULL,
  min_cells = 10L,
  min_subtypes = 2L,
  min_cpm = 1,
  min_donors = 2L,
  tc_thresh_high = 0.8,
  tc_thresh_low = 0.2,
  assay_name = "counts",
  BPPARAM = SerialParam()
)
```

Arguments

sce	A SingleCellExperiment with raw counts in assay(sce, "counts").
broad_type	A character(1) value present in colData(sce)[[broad_col]] specifying the broad population to decompose (e.g. "T_cell").
subtype_col	A character(1) naming the column in colData(sce) with fine-grained subtype labels (e.g. "cell_subtype").
broad_col	A character(1) naming the column with broad cell type labels (e.g. "cell_type").
donor	A character(1) naming the donor column.
condition	A character(1) naming the condition column. Must have exactly two unique values.
contrast	A character(1) limma contrast string, e.g. "treat - ctrl". If NULL, uses level 2 minus level 1. Default: NULL.
min_cells	An integer(1) minimum cells per donor-condition per subtype. Samples below this are removed. Default: 10L.
min_subtypes	An integer(1) minimum number of subtypes required for decomposition. Default: 2L.
min_cpm	A numeric(1) minimum CPM for gene expression filter. Default: 1.
min_donors	An integer(1) minimum number of pseudo-bulk samples a gene must be expressed in. Default: 2L.
tc_thresh_high	A numeric(1) TC_ratio threshold above which a gene is classified as *transcriptional*. Default: 0.8.
tc_thresh_low	A numeric(1) TC_ratio threshold below which a gene is classified as *compositional*. Default: 0.2.
assay_name	A character(1) name of the count assay. Default: "counts".
BPPARAM	A BiocParallelParam object. Default: SerialParam().

Details

The decomposition algorithm:

1. Filter sparse donor-subtype combinations (`min_cells`).
2. Compute subtype proportion matrix $\pi_{d,k,c}$ (donor x subtype x condition).
3. For each subtype k, aggregate pseudo-bulk and run `limma::voom` to obtain per-subtype log-fold-changes $\text{logFC}_{g,k}$.
4. Run a broad pseudo-bulk DE (all subtypes collapsed) to obtain the observed logFC_g, P-values, and FDR.
5. Compute the **transcriptional component**:

$$T_g = \sum_k \bar{\pi}_k \cdot \text{logFC}_{g,k}$$

   where $\bar{\pi}_k$ is the mean proportion of subtype k across all samples.

6. Compute the **compositional component**:

$$C_g = \sum_k \Delta\pi_k \cdot \bar{\mu}_{g,k}$$

   where $\Delta\pi_k = \bar{\pi}_{k,\text{treat}} - \bar{\pi}_{k,\text{ctrl}}$ and $\bar{\mu}_{g,k}$ is the mean log2 CPM of gene g in subtype k.

7. Z-score normalise both T_g and C_g across genes, then compute:

$$\text{TC\_ratio}_g = \frac{|T_g^z|}{|T_g^z| + |C_g^z| + \varepsilon}$$

8. Classify each gene as *transcriptional* ($\geq$ `tc_thresh_high`), *compositional* ($\leq$ `tc_thresh_low`), or *mixed*.

Value

A `CDEResult` object containing:

- `deTable`: per-gene statistics – `logFC`, `P.Value`, `adj.P.Val` (from the broad model) plus `T_score`, `C_score`, `T_score_z`, `C_score_z`, `TC_ratio`, and `source`.
- `subtypeProportions`: the π matrix.
- `subtypeDE`: per-subtype limma results.
- `params`: the analysis parameters.

See Also

`plotDecomposition`, `plotProportion`, `plotTCRatio`, `CDEResult`

Examples

```
library(SingleCellExperiment)

set.seed(42)
n_genes <- 150
n_cells <- 120

counts <- matrix(rpois(n_genes * n_cells, 8L),
                 nrow = n_genes, ncol = n_cells)
rownames(counts) <- paste0("Gene", seq_len(n_genes))
colnames(counts) <- paste0("Cell", seq_len(n_cells))

# Inject DE signal into first 10 genes for subtype A treatment
counts[seq_len(10), 61:90] <- counts[seq_len(10), 61:90] * 4L

sce <- SingleCellExperiment(assays = list(counts = counts))
sce$donor     <- rep(paste0("D", seq_len(6)), each = 20)
sce$cell_type <- "T_cell"
sce$subtype   <- rep(c("TypeA", "TypeB"), times = 60)
sce$condition <- rep(c("ctrl", "treat"), each = 60)

result <- compoundDE(
    sce,
    broad_type   = "T_cell",
    subtype_col  = "subtype",
    broad_col    = "cell_type",
    donor        = "donor",
    condition    = "condition",
    min_cells    = 3L,
    min_subtypes = 2L,
    min_donors   = 2L
)
result
head(as.data.frame(deTable(result)))
```

`deTable` *Accessor for the DE table in a CDEResult*

Description

Returns the per-gene compound DE statistics from a `CDEResult` object, including the decomposition scores.

Usage

```
deTable(x, ...)

## S4 method for signature 'CDEResult'
deTable(x, ...)
```

Arguments

x	A CDEResult object.
...	Additional arguments (not used).

Value

A DataFrame with columns logFC, P.Value, adj.P.Val, T_score, C_score, TC_ratio, and source.

Examples

```
library(S4Vectors)
dt <- DataFrame(gene = "G1", logFC = 1.2, P.Value = 0.001,
                adj.P.Val = 0.01, AveExpr = 3.1, t = 4.1, B = 2.1,
                T_score = 1.1, C_score = 0.1,
                T_score_z = 1.5, C_score_z = 0.2,
                TC_ratio = 0.88, source = "transcriptional")
pm <- matrix(c(0.6, 0.4), nrow = 1,
             dimnames = list("D1___ctrl", c("TypeA", "TypeB")))
obj <- CDEResult(dt, pm, list(), list(broad_type = "T_cell"))
deTable(obj)
```

---

filterGenesBySource *Filter DE Genes by Source Classification*

Description

A convenience utility to extract subsets of the DE table from a CDEResult object by source classification ("transcriptional", "compositional", or "mixed") and optionally by FDR significance.

Usage

```
filterGenesBySource(
  result,
  source = "transcriptional",
  fdr_thresh = 0.05,
  sort_by = "adj.P.Val"
)
```

Arguments

result	A CDEResult object.
source	A character vector of source categories to retain. Must be one or more of "transcriptional", "compositional", "mixed". Default: "transcriptional".
fdr_thresh	A numeric(1) FDR threshold. Only genes with adj.P.Val < fdr_thresh are returned. Set to 1 to return all genes of the given source. Default: 0.05.
sort_by	A character(1) column to sort results by. Default: "adj.P.Val".

Value

A `data.frame` of filtered DE genes.

See Also

`compoundDE`, `deTable`

Examples

```
library(SingleCellExperiment)

set.seed(42)
n_genes <- 150
counts <- matrix(rpois(n_genes * 120, 8L), nrow = n_genes, ncol = 120)
rownames(counts) <- paste0("Gene", seq_len(n_genes))
colnames(counts) <- paste0("Cell", seq_len(120))
counts[seq_len(10), 61:90] <- counts[seq_len(10), 61:90] * 4L

sce <- SingleCellExperiment(assays = list(counts = counts))
sce$donor     <- rep(paste0("D", seq_len(6)), each = 20)
sce$cell_type <- "T_cell"
sce$subtype   <- rep(c("TypeA", "TypeB"), times = 60)
sce$condition <- rep(c("ctrl", "treat"), each = 60)

result <- compoundDE(sce, broad_type = "T_cell",
    subtype_col = "subtype", broad_col = "cell_type",
    donor = "donor", condition = "condition",
    min_cells = 3L, min_subtypes = 2L, min_donors = 2L)

# Get only transcriptionally significant genes
filterGenesBySource(result, source = "transcriptional")

# Get compositional artefacts
filterGenesBySource(result, source = "compositional", fdr_thresh = 1)
```

---

plotDecomposition *Scatter Plot of Transcriptional vs Compositional DE Scores*

Description

Produces a scatter plot of the z-scored transcriptional score (`T_score_z`) versus the compositional score (`C_score_z`) for every gene, coloured by their `source` classification. This visualisation makes it immediately clear which genes are driven by cell-intrinsic expression changes versus subtype proportion shifts.

Usage

```
plotDecomposition(
  result,
  fdr_thresh = 0.05,
  top_n = 10L,
  point_size = 1.2,
  point_alpha = 0.7,
 colours = c(transcriptional = "#2196F3", compositional = "#F44336", mixed = "#FF9800",
    ns = "#BDBDBD")
)
```

Arguments

result	A CDEResult object from compoundDE.
fdr_thresh	A numeric(1) FDR threshold. Only genes with adj.P.Val < fdr_thresh are highlighted as significant. Default: 0.05.
top_n	An integer(1) number of top significant genes to label by name. Default: 10L.
point_size	A numeric(1) point size. Default: 1.2.
point_alpha	A numeric(1) point transparency. Default: 0.7.
colours	A named character vector with colours for "transcriptional", "compositional", "mixed", and "ns" (not significant). Default uses a colour-blind friendly palette.

Value

A ggplot2 object.

See Also

compoundDE, plotTCRatio, plotProportion

Examples

```
library(SingleCellExperiment)

set.seed(42)
n_genes <- 150
counts <- matrix(rpois(n_genes * 120, 8L), nrow = n_genes, ncol = 120)
rownames(counts) <- paste0("Gene", seq_len(n_genes))
colnames(counts) <- paste0("Cell", seq_len(120))
counts[seq_len(10), 61:90] <- counts[seq_len(10), 61:90] * 4L

sce <- SingleCellExperiment(assays = list(counts = counts))
sce$donor     <- rep(paste0("D", seq_len(6)), each = 20)
sce$cell_type <- "T_cell"
sce$subtype   <- rep(c("TypeA", "TypeB"), times = 60)
sce$condition <- rep(c("ctrl", "treat"), each = 60)

result <- compoundDE(sce, broad_type = "T_cell",
    subtype_col = "subtype", broad_col = "cell_type",
```

```
    donor = "donor", condition = "condition",
    min_cells = 3L, min_subtypes = 2L, min_donors = 2L)
plotDecomposition(result)
```

`plotProportion` *Stacked Bar Plot of Subtype Proportions per Condition*

Description

Produces a stacked bar chart showing the mean proportion of each subtype within the broad cell type, split by condition. Samples are shown as individual points overlaid on the bars, making it easy to see between-donor variability in composition. A significant compositional shift across conditions is a key driver of *compositional* DE genes identified by `compoundDE`.

Usage

```
plotProportion(
  result,
  show_points = TRUE,
  colours = NULL,
  point_size = 2,
  point_alpha = 0.8
)
```

Arguments

`result`	A `CDEResult` object from `compoundDE`.
`show_points`	Logical. If `TRUE`, overlay individual sample proportions as jittered points. Default: `TRUE`.
`colours`	A named or unnamed character vector of colours for the subtypes. If `NULL`, a default palette is used. Default: `NULL`.
`point_size`	A `numeric(1)` point size. Default: `2`.
`point_alpha`	A `numeric(1)` point alpha. Default: `0.8`.

Value

A `ggplot2` object.

See Also

`compoundDE`, `plotDecomposition`, `plotTCRatio`

Examples

```
library(SingleCellExperiment)

set.seed(42)
n_genes <- 150
counts <- matrix(rpois(n_genes * 120, 8L), nrow = n_genes, ncol = 120)
rownames(counts) <- paste0("Gene", seq_len(n_genes))
colnames(counts) <- paste0("Cell", seq_len(120))
counts[seq_len(10), 61:90] <- counts[seq_len(10), 61:90] * 4L

sce <- SingleCellExperiment(assays = list(counts = counts))
sce$donor     <- rep(paste0("D", seq_len(6)), each = 20)
sce$cell_type <- "T_cell"
sce$subtype   <- rep(c("TypeA", "TypeB"), times = 60)
sce$condition <- rep(c("ctrl", "treat"), each = 60)

result <- compoundDE(sce, broad_type = "T_cell",
    subtype_col = "subtype", broad_col = "cell_type",
    donor = "donor", condition = "condition",
    min_cells = 3L, min_subtypes = 2L, min_donors = 2L)
plotProportion(result)
```

---

plotTCRatio *Histogram of TC_ratio Distribution*

Description

Plots the distribution of `TC_ratio` values across all tested genes. The TC_ratio measures the fraction of each gene's DE signal that is attributable to transcriptional (cell-intrinsic) versus compositional (subtype proportion shift) change. Vertical dashed lines show the classification thresholds.

Usage

```
plotTCRatio(
  result,
  fdr_thresh = 0.05,
  bins = 40L,
 colours = c(transcriptional = "#2196F3", compositional = "#F44336", mixed = "#FF9800",
    ns = "#BDBDBD")
)
```

Arguments

`result`	A `CDEResult` object from `compoundDE`.
`fdr_thresh`	A `numeric(1)` FDR threshold. Genes with `adj.P.Val >= fdr_thresh` are shown in grey. Default: `0.05`.
`bins`	An `integer(1)` number of histogram bins. Default: `40L`.

colours — A named character vector with colours for "transcriptional", "compositional", "mixed", and "ns". Default uses a colour-blind friendly palette.

Value

A ggplot2 object.

See Also

compoundDE, plotDecomposition, plotProportion

Examples

```
library(SingleCellExperiment)

set.seed(42)
n_genes <- 150
counts <- matrix(rpois(n_genes * 120, 8L), nrow = n_genes, ncol = 120)
rownames(counts) <- paste0("Gene", seq_len(n_genes))
colnames(counts) <- paste0("Cell", seq_len(120))
counts[seq_len(10), 61:90] <- counts[seq_len(10), 61:90] * 4L

sce <- SingleCellExperiment(assays = list(counts = counts))
sce$donor     <- rep(paste0("D", seq_len(6)), each = 20)
sce$cell_type <- "T_cell"
sce$subtype   <- rep(c("TypeA", "TypeB"), times = 60)
sce$condition <- rep(c("ctrl", "treat"), each = 60)

result <- compoundDE(sce, broad_type = "T_cell",
    subtype_col = "subtype", broad_col = "cell_type",
    donor = "donor", condition = "condition",
    min_cells = 3L, min_subtypes = 2L, min_donors = 2L)
plotTCRatio(result)
```

---

show,CDEResult-method *Show method for CDEResult*

Description

Prints a compact summary of a CDEResult object including the decomposition breakdown.

Usage

```
## S4 method for signature 'CDEResult'
show(object)
```

Arguments

object — A CDEResult object.

Value

Invisibly returns `object`.

`subtypeDE` *Accessor for per-subtype DE results in a CDEResult*

Description

Returns the list of per-subtype limma DE `DataFrames`.

Usage

```
subtypeDE(x, ...)

## S4 method for signature 'CDEResult'
subtypeDE(x, ...)
```

Arguments

`x`	A `CDEResult` object.
`...`	Additional arguments (not used).

Value

A named `list` of `DataFrames`, one per subtype.

Examples

```
library(S4Vectors)
dt <- DataFrame(gene = "G1", logFC = 1.2, P.Value = 0.001,
                adj.P.Val = 0.01, AveExpr = 3.1, t = 4.1, B = 2.1,
                T_score = 1.1, C_score = 0.1,
                T_score_z = 1.5, C_score_z = 0.2,
                TC_ratio = 0.88, source = "transcriptional")
pm <- matrix(c(0.6, 0.4), nrow = 1,
             dimnames = list("D1___ctrl", c("TypeA", "TypeB")))
obj <- CDEResult(dt, pm, list(), list(broad_type = "T_cell"))
subtypeDE(obj)
```

subtypeProportions *Accessor for subtype proportions in a CDEResult*

Description

Returns the subtype proportion matrix from a `CDEResult` object.

Usage

```
subtypeProportions(x, ...)

## S4 method for signature 'CDEResult'
subtypeProportions(x, ...)
```

Arguments

`x`	A `CDEResult` object.
`...`	Additional arguments (not used).

Value

A `matrix` with rows = samples and columns = subtypes.

Examples

```
library(S4Vectors)
dt <- DataFrame(gene = "G1", logFC = 1.2, P.Value = 0.001,
                adj.P.Val = 0.01, AveExpr = 3.1, t = 4.1, B = 2.1,
                T_score = 1.1, C_score = 0.1,
                T_score_z = 1.5, C_score_z = 0.2,
                TC_ratio = 0.88, source = "transcriptional")
pm <- matrix(c(0.6, 0.4), nrow = 1,
             dimnames = list("D1___ctrl", c("TypeA", "TypeB")))
obj <- CDEResult(dt, pm, list(), list(broad_type = "T_cell"))
subtypeProportions(obj)
```

tcRatio *Accessor for TC_ratio vector in a CDEResult*

Description

Returns the per-gene TC_ratio vector – the fraction of DE signal attributable to transcriptional versus compositional change. Values near 1 = transcriptional (real biology); values near 0 = compositional (artifact).

Usage

```
tcRatio(x, ...)

## S4 method for signature 'CDEResult'
tcRatio(x, ...)
```

Arguments

x	A CDEResult object.
...	Additional arguments (not used).

Value

A named numeric vector of TC_ratio values.

Examples

```
library(S4Vectors)
dt <- DataFrame(gene = c("G1","G2"), logFC = c(1.2,0.1),
                P.Value = c(0.001,0.8), adj.P.Val = c(0.01,0.9),
                AveExpr = c(3.1,2.0), t = c(4.1,0.5), B = c(2.1,-2.0),
                T_score = c(1.1,0.05), C_score = c(0.1,0.09),
                T_score_z = c(1.5,0.2), C_score_z = c(0.2,0.3),
                TC_ratio = c(0.88,0.40), source = c("transcriptional","mixed"))
pm <- matrix(c(0.6,0.4,0.3,0.7), nrow=2,
             dimnames=list(c("D1___ctrl","D1___treat"),c("TypeA","TypeB")))
obj <- CDEResult(dt, pm, list(), list(broad_type = "T_cell"))
tcRatio(obj)
```

Index