Package 'MICSQTL'

November 12, 2026

Type Package

Title MICSQTL (Multi-omic deconvolution, Integration and Cell-type-specific Quantitative Trait Loci)

Version 1.10.0

Description Our pipeline, MICSQTL, utilizes scRNA-seq reference and bulk transcriptomes to estimate cellular composition in the matched bulk proteomes. The expression of genes and proteins at either bulk level or cell type level can be integrated by Angle-based Joint and Individual Variation Explained (AJIVE) framework. Meanwhile, MICSQTL can perform cell-type-specic quantitative trait loci (QTL) mapping to proteins or transcripts based on the input of bulk expression data and the estimated cellular composition per molecule type, without the need for single cell sequencing. We use matched transcriptome-proteome from human brain frontal cortex tissue samples to demonstrate the input and output of our tool.

License GPL-3

Encoding UTF-8

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

VignetteBuilder knitr

biocViews GeneExpression, Genetics, Proteomics, RNASeq, Sequencing, SingleCell, Software, Visualization, CellBasedAssays, Coverage

Imports TCA, nnls, purrr, TOAST, magrittr, BiocParallel, ggplot2, ggpubr, ggridges, glue, S4Vectors, dirmult

Depends R (>= 4.3.0), SummarizedExperiment, stats

RoxygenNote 7.2.3

LazyData false

Config/testthat/edition 3

URL https://bioconductor.org/packages/MICSQTL, https://github.com/YuePan027/MICSQTL

BugReports https://github.com/YuePan027/MICSQTL/issues

git_url https://git.bioconductor.org/packages/MICSQTL

git_branch RELEASE_3_23

git_last_commit 8a615ba

git_last_commit_date 2026-04-28

Repository Bioconductor 3.23

Date/Publication 2026-11-11

Author Yue Pan [aut] (ORCID: <https://orcid.org/0000-0003-1958-2744>),
Qian Li [aut, cre] (ORCID: <https://orcid.org/0000-0001-9874-3555>),
Iain Carmichael [ctb]

Maintainer Qian Li <qian.li@stjude.org>

Contents

ajive_decomp 2
csQTL 4
deconv 5
feature_filter 7
se 9

Index **10**

`ajive_decomp` *Integrative analysis for modes of joint variation*

Description

This function returns a 'SummarizedExperiment' object including results from AJIVE (Angle based Joint and Individual Variation Explained), an integrative analysis tool, and common normalized scored based on it. This implemented the AJIVE algorithm (see R version of AJIVE package (https://github.com/idc9/r_jive) or detailed manuscript at https://arxiv.org/pdf/1704.02060.pdf) an integrative analysis method. It is useful when there are multiple data matrices measured on the same set of samples. It decomposes each data matrix as three parts: (1) Joint variation across data types (2) Individual structured variation for each data type and (3) Residual noise. Common normalized scores are one of the desirable output to explore the joint behavior that is shared by different data sources.

Usage

```
ajive_decomp(
  se,
  ini_rank = c(20, 20),
  test = "gene_data",
  use_marker = FALSE,
  level = "bulk",
  plot = FALSE,
  score = "cns_1",
  group_var = "disease",
  scatter = FALSE,
  scatter_x,
```

```
  scatter_y,
  refactor_loading = FALSE
)
```

Arguments

se	A 'SummarizedExperiment' object with bulk expression data frame contained in 'counts' slot. In addition, data measured on the same set of samples in 'metadata' slot is required.
ini_rank	A vector with each element corresponds to a initial signal rank for AJIVE decomposition. Please refer to the original paper (Feng, Qing, et al. "Angle-based joint and individual variation explained." Journal of multivariate analysis 166 (2018): 241-265.) on the choice of initial ranks.
test	A character string indicate which data are used as a secondary data block measured on the same set of samples.
use_marker	If TRUE, only markers contained in 'ref_gene' are used.
level	A character string indicate if the integrative analysis should be done at cell type specific level and which cell type should be used. By default, the integrative analysis is done at bulk level.
plot	If TRUE, visualization on common normalized scores across different data sources will be stored as an element ('cns_plot') in 'metadata' slot.
score	A character of variable name indicating which common normalized score is used for boxplot and ridge plot (valid if plot = TRUE).
group_var	A character of variable name indicating which variable is used as the group variable to compare common normalized scores (valid if plot = TRUE).
scatter	A logical value indicating whether to make scatter plot or not. Only valid when the joint rank is at least two (valid if plot = TRUE).
scatter_x	A character of variable name indicating which common normalized scores on horizontal axis (valid if plot = TRUE).
scatter_y	A character of variable name indicating which common normalized scores on vertical axis (valid if plot = TRUE).
refactor_loading	A logical value indicating whether to output the refactor joint loadings for each data source. A smaller value indicates features with the highest variance explained within the joint space.

Value

A 'SummarizedExperiment'. The results from AJIVE will be stored as an element ('ajive_res') in 'metadata' slot. The generated common normalized scores will be stored as an element ('cns') in 'metadata' slot. The visualization of the common normalized scores will be stored as an element ('cns_plot') in 'metadata' slot.

Examples

```
data(se)
metadata(se)$gene_data <- metadata(se)$gene_data[seq_len(100),] #reduce time
se <- ajive_decomp(se, use_marker = TRUE)
```

---

csQTL *Cell-type-specific differential expression (csQTL)*

Description

This function returns a 'SummarizedExperiment' object including csQTL proteins based on samples' genotype.

Usage

```
csQTL(se, BPPARAM = bpparam())
```

Arguments

se	A 'SummarizedExperiment' object with bulk protein expression data frame contained in 'counts' slot. The information from genetic variants should be stored in a P (the number of SNP) by N (the number of samples, should match the sample in 'counts' slot) matrix contained as an element ('SNP_data') in 'metadata' slot. Each matrix entry corresponds to the genotype group indicator (0, 1 or 2) for a sample at a genetic location. The annotations of these SNP should be stored as an element ('anno_SNP') in 'metadata' slot. It should include at least the following columns: "CHROM" (which chromosome the SNP is on), "POS" (position of that SNP) and "ID" (a unique identifier for each SNP, usually a combination of chromosome and its position). The information on cellular composition is required and stored as 'prop' in 'metadata' slot. It is an N (the number of samples, should match the sample in 'counts' slot) by K (the number of cell types) matrix. This can be obtained by running 'deconv()' before any filtering steps, or use any source data directly.
BPPARAM	For applying 'bplapply'.

Details

This is a function developed to implement cell-type-specific differential expression using 'TOAST'.

Value

A 'SummarizedExperiment'. The csQTL results will be stored as an element ('TOAST_output') in 'metadata' slot.

Examples

```
data(se)
se <- deconv(se, source = "protein", method = "nnls", use_refactor = NULL)
target_protein <- c("ABCA1")
se <- feature_filter(se,
    target_protein = target_protein,
    filter_method = c("allele", "distance"), filter_allele = 0.15,
    filter_geno = 0.05, ref_position = "TSS"
)
se <- csQTL(se)
```

---

deconv *Estimation of cellular composition in high-throughput data from heterogeneous tissues*

Description

This function returns a 'SummarizedExperiment' object including cell-type proportion estimates for each sample.

Usage

```
deconv(
  se,
  source = "cross",
  method = c("nnls", "JNMF", "TCA"),
  use_refactor = c(1000, NULL),
  Step = c(10^(-8), 10^(-6)),
  Eps = 10^(-4),
  Iter = 500,
  cell_counts = NULL,
  pinit = "nnls",
  ref_pnl = NULL
)
```

Arguments

se	A 'SummarizedExperiment' object with bulk protein expression data frame contained in assay, a bulk transcript expression data frame ('gene_data') contained in 'metadata' slot (can be rescaled using either log or MinMax transformations, but need to be consistent between two bulk data). A "signature matrix" functions as a reference containing known cellular signatures (either 'ref_protein' or 'ref_gene' as an element in the 'metadata' slot) may be necessary for certain 'source'and 'method' options. To ensure the reliability of the results obtained, we strongly recommend that the "signature matrix" should exclusively comprise markers that have been previously validated in the literature.

`source` A character string denotes which molecular profiles to be deconvoluted. The setting of ‘proteins‘or ‘transcript‘ means single-source deconvolution with source-specific signature matrix, while ‘cross‘ means proteome deconvolution based on transcriptome-proteome with matched samples.

`method` A character string specifies the deconvolution method to be employed. In the current version, only ’nnls’ is supported for single-source deconvolution. Besides the bulk data, ‘ref_protein‘ or ‘ref_gene‘ is required for protein or gene deconvolution, respectively. For cross-source deconvolution, ‘JNMF‘ or ‘TCA‘ are valid options. If ‘JNMF‘, an external reference containing cell counts in a similar tissue type (typically obtainable from small-scale single-cell or flow cytometry experiments) is necessary if ‘pinit = "rdirichlet"‘; A "signature matrix" is required for other methods. If ‘TCA‘, an input of pre-estimated transcriptome proportions, denoted as ‘prop_gene‘ as an element in the ‘metadata‘ slot, is required. This input can be derived from single-source deconvolution using ‘nnls‘ included in this package, or from an external source.

`use_refactor` A numeric value indicate the number of proteins included for proportion estimates based on refactor values. Note that ‘ajive_decomp‘ with ‘refactor_loading = TRUE‘ required if this method applied. If NULL, then all proteins included in assay will be used.

`Step` A numeric vector indicates the step size in projected gradient descent for cell count fraction parameter and cell size parameter, respectively. Only valid if ‘method = JNMF‘.

`Eps` A numeric value indicates the convergence criteria for projected gradient descent. Only valid if ‘method = JNMF‘.

`Iter` A numeric value indicates the maximum iteration time for projected gradient descent. Only valid if ‘method = JNMF‘.

`cell_counts` A matrix containing cell counts across multiple subjects, where subjects are represented as rows and cell types as columns. Each entry (i, j) in the matrix indicates the count of cells belonging to the ith subject and jth cell type. Only required if ‘method = JNMF‘ and ‘pinit = "rdirichlet"‘.

`pinit` Accepts either a numeric matrix or a character indicating the method for initializing initial values for cellular fraction. If ‘pinit‘ is a numeric matrix (pre-estimated transcriptome proportions using other methods), each row represents the cellular fraction for each sample across various cell types. The resulting cellular fraction will match the cell types defined in ‘pinit‘. Alternatively, ‘pinit‘ can be generated using either the ‘rdirichlet‘ or ‘nnls‘ method.

`ref_pnl` A "signature matrix" functions as a reference containing known cellular signatures. It is optional. If provided, the initial values for purified data will be generated based on ‘ref_pnl‘. Otherwise, the initial values for purified data will be generated using a normal distribution based on bulk data. Please note that the input signature matrix should have the same rescaling transformation as the bulk transcriptomes/proteomic.

Details

This is a function developed to implement cell-type proportion deconvolution using either single or cross sources.

Value

A 'SummarizedExperiment'. The cell-type proportion estimates for each sample are stored as elements starting with prop in the metadata slot. If 'method = JNMF', then the cellular fractions obtained from proteomics and transcriptomics are stored in the 'prop' and 'prop2' elements, respectively, within the metadata slot. The purified data is stored in a list with the same length as the number of subjects (the number of columns in the assay). For subject i, the purified protein expression data can be obtained by accessing 'se_sim@metadata$purified[[i]][["X1"]]', and similarly, the purified transcript expression data can be obtained by accessing 'se_sim@metadata$purified[[i]][["X2"]'].

Examples

```
data(se)
se <- deconv(se, source = "protein", method = "nnls", use_refactor = NULL)
```

`feature_filter` *Feature filtering*

Description

This function returns a 'SummarizedExperiment' object including SNPs used to test for each protein in downstream analysis.

Usage

```
feature_filter(
  se,
  target_protein = NULL,
  target_SNP = NULL,
  filter_method = c("allele", "distance", "null"),
  filter_allele = 0.25,
  filter_geno = 0.05,
  ref_position = c("TSS", "genebody"),
  BPPARAM = bpparam()
)
```

Arguments

`se`	A 'SummarizedExperiment' object with bulk protein expression data frame contained in 'counts' slot. Annotations on each row (protein) should be stored in rowData() with protein symbol as row names The first column should be a character vector indicating which chromosome each protein is on. A "Start" column with numeric values indicating the start position on that chromosome and a "Symbol" column as a unique name for each protein is also required. The information from genetic variants should be stored in a P (the number of SNP) by N (the number of samples, should match the sample in 'counts' slot) matrix contained as an element ('SNP_data') in 'metadata' slot. Each matrix

entry corresponds to the genotype group indicator (0, 1 or 2) for a sample at a genetic location. The annotations of these SNP should be stored as an element (‘anno_SNP‘) in ‘metadata‘ slot. It should include at least the following columns: "CHROM" (which chromosome the SNP is on), "POS" (position of that SNP) and "ID" (a unique identifier for each SNP, usually a combination of chromosome and its position).

`target_protein` A character vector contains proteins names that will be used for downstream analysis. By default, all proteins in ‘counts‘ slot will be used.

`target_SNP` A character vector contains SNP IDs that will be used for downstream analysis. If not provided, all SNPs will be used for further filtering.

`filter_method` A character string denotes which filtering method will be used to filter out unrelated SNPs. If "allele", then the minor allele frequency below argument ‘filter_allele‘ will be filtered out. If "distance", then only cis-acting SNPs for each protein (defined as SNPs on the same chromosome and within 1M base pair (bp) range of that protein) will be included for downstream analysis. if "null", then the same SNPs will be used for each protein.

`filter_allele` A numeric value denotes the threshold for minor allele frequency. Only works when ‘filter_method‘ contains "allele".

`filter_geno` A numeric value denotes the threshold for minimum genotype group proportion. Only works when ‘filter_method‘ contains "allele".

`ref_position` A character string denotes the reference position on protein when ‘filter_method‘ contains "distance", where "TSS" refers to transcription start site, and "genebody" refers to the middle point of "Start" and "End" position.

`BPPARAM` For applying ‘bplapply‘.

Details

This is a function developed to filter unwanted proteins or SNPs with less variation among samples for downstream analysis.

Value

A ‘SummarizedExperiment‘. The results after filtering will be stored as an element (‘choose_SNP_list‘) in ‘metadata‘ slot. ‘choose_SNP_list‘ is a list with the length of the number of proteins for downstream analysis. Each element stores the index of SNPs to be tested for corresponding protein. The proteins with no SNPs correspond to it will be removed from the returned list.

Examples

```
data(se)
target_protein <- rowData(se)[rowData(se)$Chr == 9, ][seq_len(3), "Symbol"]
se <- feature_filter(se,
    target_protein = target_protein,
    filter_method = c("allele", "distance"),
    filter_allele = 0.15, filter_geno = 0.05, ref_position = "TSS"
)
```

se *Example data*

Description

The example input files are organized as a 'SummarizedExperiment'object.

Format

A 'SummarizedExperiment'object with following example data:

protein_data An example proteomics data (on log scale) with 2,242 rows (protein) and 127 columns (sample).

anno_protein A data frame with 2242 rows and 4 columns (Chr, Start, End, Symbol) as annotations of each protein from 'protein_data'.

ref_protein A signature matrix with 2242 rows (protein) and 4 columns (cell types), which serves as a reference of known cellular signatures.

gene_data A data frame with 2867 rows (genes) and 127 columns (sample).

prop_gene A pre-defined deconvoluted transcriptome proportion matrix.

ref_gene A signature matrix with 4872 rows (genes) and 5 columns (cell types), which serves as a reference of known cellular signatures.

SNP_data A sparse matrix with 2000 rows (SNP), which stores the information of genetic variants at each location from one chromosome and 127 columns (sample, should match the sample in 'protein_data'). Each matrix entry corresponds to the genotype group indicator (0, 1 or 2) for a sample at a genetic location.

anno_SNP A data frame with 2000 rows and 3 columns (CHROM, POS, ID), which stores Annotations of each SNP from 'SNP_data'

meta A data frame with 127 rows (sample) and 2 columns (disease status and gender) as metadata.

prop An example cellular composition by running 'deconv' function.

cell_counts A matrix containing cell counts across multiple subjects, where subjects are represented as rows and cell types as columns. Each entry (i, j) in the matrix indicates the count of cells belonging to the ith subject and jth cell type.

Value

A 'SummarizedExperiment'object.

Examples

```
data(se)
```

Index

∗ **datasets**
    se, 9

ajive_decomp, 2

csQTL, 4

deconv, 5

feature_filter, 7

se, 9