Package 'quantiseqr'

November 9, 2026

Title Quantification of the Tumor Immune contexture from RNA-seq data

Version 1.20.0

Description This package provides a streamlined workflow for the quanTIseq method, developed to perform the quantification of the Tumor Immune contexture from RNA-seq data. The quantification is performed against the TIL10 signature (dissecting the contributions of ten immune cell types), carefully crafted from a collection of human RNA-seq samples. The TIL10 signature has been extensively validated using simulated, flow cytometry, and immunohistochemistry data.

License GPL-3

Depends R (>= 4.1.0)

Imports Biobase, limSolve, MASS, methods, preprocessCore, stats, SummarizedExperiment, ggplot2, tidyr, rlang, utils

Suggests AnnotationDbi, BiocStyle, dplyr, ExperimentHub, GEOquery, knitr, macrophage, org.Hs.eg.db, reshape2, rmarkdown, testthat, tibble

biocViews GeneExpression, Software, Transcription, Transcriptomics, Sequencing, Microarray, Visualization, Annotation, ImmunoOncology, FeatureExtraction, Classification, StatisticalMethod, ExperimentHubSoftware, FlowCytometry

VignetteBuilder knitr

Encoding UTF-8

Roxygen list(markdown = TRUE)

RoxygenNote 7.3.2

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

git_branch RELEASE_3_23

git_last_commit a948291

git_last_commit_date 2026-04-28

Repository Bioconductor 3.23

Date/Publication 2026-11-08

Author Federico Marini [aut, cre] (ORCID:
<https://orcid.org/0000-0003-3252-7758>),
Francesca Finotello [aut] (ORCID:
<https://orcid.org/0000-0003-0712-4658>)

Maintainer Federico Marini <marinif@uni-mainz.de>

Contents

check_signature 2
dataset_racle 3
DClsei 3
DCrr 4
eset_to_matrix 5
extract_ti_from_se 6
fixMixture 7
get_densities 8
makeQN 9
mapGenes 9
quantiplot 10
quanTIseq 11
quantiseqr-pkg 12
quantiseq_helper 12
run_quantiseq 12
se_to_matrix 15
ti_quant_sim1700mixtures 16

Index **17**

`check_signature` *Check the signature matrix*

Description

Checks requirements for the signature matrix, with respect to the expression matrix data provided (the one on which the deconvolution algorithm needs to be run)

Usage

```
check_signature(signature_matrix, mix_mat)
```

Arguments

`signature_matrix`	A data.frame or a matrix object, containing the signature matrix
`mix_mat`	Mixture matrix, storing the information provided as `expression_data` to the main function, `run_quantiseqr()`

Details

Performs a number of checks to ensure the compatibility of the provided signature matrix in `quantiseqr`, referring also to the content of the `mix_mat` mixture matrix, to be deconvoluted

Value

Invisible NULL

`dataset_racle` *An exemplary dataset with samples from four patients with metastatic melanoma*

Description

An exemplary dataset with samples from four patients with metastatic melanoma

Details

`quantiseqr` ships with an example dataset with samples from four patients with metastatic melanoma. The dataset `quantiseqr::dataset_racle` contains

- a gene expression matrix (`dataset_racle$expr_mat`) generated using bulk RNA-seq
- 'gold standard' estimates of immune cell contents profiled with FACS (`dataset_racle$ref`).

References

Racle et al, 2017 - https://doi.org/10.7554/eLife.26476.049

`DClsei` *Solve Least Squares with Equality and Inequality Constraints (LSEI) problem*

Description

Solve Least Squares with Equality and Inequality Constraints (LSEI) problem

Usage

```
DClsei(b, A, G, H, scaling = NULL)
```

Arguments

b	Numeric vector containing the right-hand side of the quadratic function to be minimized.
A	Numeric matrix containing the coefficients of the quadratic function to be minimized.
G	Numeric matrix containing the coefficients of the inequality constraints.
H	Numeric vector containing the right-hand side of the inequality constraints.
scaling	A vector of scaling factors to by applied to the estimates. Its length should equal the number of columns of A.

Details

The `limSolve::lsei()` function is used as underlying framework. Please refer to that function for more details.

Value

A vector containing the solution of the LSEI problem.

Examples

```
data(dataset_racle)
mixture <- dataset_racle$expr_mat
signature.file <- system.file(
  "extdata", "TIL10_signature.txt", package = "quantiseqr", mustWork = TRUE)
signature <- read.table(signature.file, header = TRUE, sep = "\t", row.names = 1)
scaling.file <- system.file(
  "extdata", "TIL10_mRNA_scaling.txt", package = "quantiseqr", mustWork = TRUE)
scaling <- as.vector(
  as.matrix(read.table(scaling.file, header = FALSE, sep = "\t", row.names = 1)))

cgenes <- intersect(rownames(signature), rownames(mixture))
b <- as.vector(as.matrix(mixture[cgenes,1, drop=FALSE]))
A <- as.matrix(signature[cgenes,])

G <- matrix(0, ncol = ncol(A), nrow = ncol(A))
diag(G) <- 1
G <- rbind(G, rep(-1, ncol(G)))
H <- c(rep(0, ncol(A)), -1)
# cellfrac <- quantiseqr:::DClsei(b = b, A = A, G= G, H = H, scaling = scaling)
```

DCrr *Perform robust regression*

Description

Perform robust regression

Usage

```
DCrr(b, A, method = c("hampel", "huber", "bisquare"), scaling = NULL)
```

Arguments

b	Numeric vector containing the right-hand side of the quadratic function to be minimized.
A	Numeric matrix containing the coefficients of the quadratic function to be minimized.
method	Character specifying the robust regression method to be used among deconvolution methods: "hampel", "huber", or "bisquare". Default: "hampel".
scaling	A vector of scaling factors to by applied to the estimates. Its length should equal the number of columns of A.

Details

The `MASS::rlm()` function is used as underlying framework. Please refer to that function for more details.

Value

A vector containing robust least-square estimates.

Examples

```
data(dataset_racle)
mixture <- dataset_racle$expr_mat
signature.file <- system.file(
  "extdata", "TIL10_signature.txt", package = "quantiseqr", mustWork = TRUE)
signature <- read.table(signature.file, header = TRUE, sep = "\t", row.names = 1)
scaling.file <- system.file(
  "extdata", "TIL10_mRNA_scaling.txt", package = "quantiseqr", mustWork = TRUE)
scaling <- as.vector(
  as.matrix(read.table(scaling.file, header = FALSE, sep = "\t", row.names = 1)))

cgenes <- intersect(rownames(signature), rownames(mixture))
b <- as.vector(as.matrix(mixture[cgenes,1, drop=FALSE]))
A <- as.matrix(signature[cgenes,])

# cellfrac <- quantiseqr:::DCrr(b = b, A = A, scaling = scaling)
```

`eset_to_matrix` *Convert a* `Biobase::ExpressionSet` *to a gene-expression matrix.*

Description

Convert a `Biobase::ExpressionSet` to a gene-expression matrix.

Usage

```
eset_to_matrix(eset, column)
```

Arguments

eset	ExpressionSet
column	column name of the fData() table, which contains the HGNC gene symbols.

Value

A matrix with gene symbols as rownames and sample identifiers as colnames.

Examples

```
data(dataset_racle)
dim(dataset_racle$expr_mat)

library("Biobase")
es_racle <- ExpressionSet(assayData = dataset_racle$expr_mat)
featureData(es_racle)$gene_symbol <- rownames(dataset_racle$expr_mat)

es_racle

head(eset_to_matrix(es_racle, "gene_symbol"))
```

---

extract_ti_from_se *Extract tumor immune quantifications*

Description

Extract tumor immune quantifications from a SummarizedExperiment object, previously processed with run_quantiseqr()

Usage

```
extract_ti_from_se(se)
```

Arguments

se	A SummarizedExperiment object, or any of its derivates, which contains the quantifications extracted via quantiseqr in its colData slot.

Value

A data.frame, formatted as required by downstream functions

Examples

```
data(dataset_racle)
dim(dataset_racle$expr_mat)

# using a SummarizedExperiment object
library("SummarizedExperiment")
se_racle <- SummarizedExperiment(
  assays = List(
    abundance = dataset_racle$expr_mat
  ),
  colData = DataFrame(
    SampleName = colnames(dataset_racle$expr_mat)
  )
)

res_run_SE <- quantiseqr::run_quantiseq(
    expression_data = se_racle,
    signature_matrix = "TIL10",
    is_arraydata = FALSE,
    is_tumordata = TRUE,
    scale_mRNA = TRUE
)

extract_ti_from_se(res_run_SE)
```

`fixMixture` *Format the mixture matrix before deconvolution*

Description

Format the mixture matrix before deconvolution

Usage

```
fixMixture(mix.mat, arrays = FALSE)
```

Arguments

`mix.mat`	Matrix or data.frame with RNA-seq gene TPM or microarray expression values for all samples to be deconvoluted, with gene symbols as row names and sample IDs as column names. Expression levels should be on non-log scale.
`arrays`	Logical value. Should be set to TRUE if the expression data are from microarrays. For RNA-seq data, this has to be FALSE (default value).

Value

The input matrix transformed to the natural scale (if needed), with fixed gene names on the rows, and TPM (for RNA-seq) or quantile (for microarrays) normalized.

Examples

```
data(dataset_racle)
# mixture.fix <- quantiseqr:::fixMixture(dataset_racle$expr_mat)
```

`get_densities` *Scale deconvoluted cell fractions to cell densities*

Description

Scale deconvoluted cell fractions to cell densities

Usage

```
get_densities(DCres, density_info)
```

Arguments

`DCres`	Data.frame of deconvoluted cell fractions computed with the `run_quantiseq()` function, with sample identifiers as row names.
`density_info`	Named numeric vector of total cell densities per sample. The vector names should match the sample identifiers specified in DCres. These values are derived from the quantitative analysis of imaging data.

Value

A data.frame of cell densities, samples by cell types.

Examples

```
data(dataset_racle)
mixture <- dataset_racle$expr_mat

res_quantiseq_run <- quantiseqr::run_quantiseq(
  expression_data = dataset_racle$expr_mat,
  signature_matrix = "TIL10",
  is_arraydata = FALSE,
  is_tumordata = TRUE,
  scale_mRNA = TRUE
)

totcells <- rnorm(n = ncol(mixture), mean = 1e4)
names(totcells) <- colnames(mixture)
celldens <- get_densities(res_quantiseq_run, totcells)
```

makeQN *Perform quantile normalization of expression data*

Description

Perform quantile normalization of expression data

Usage

```
makeQN(mix.mat)
```

Arguments

mix.mat	Matrix or data.frame with microarray gene expression values for all samples to be deconvoluted, with gene symbols as row names and sample IDs as column names. Expression levels should be on non-log scale.

Value

The input matrix transformed with quantile normalization.

Examples

```
data(dataset_racle)
# mixture.quantile <- quantiseqr:::makeQN(dataset_racle$expr_mat)
```

mapGenes *Rename gene symbols before deconvolution*

Description

Rename gene symbols before deconvolution

Usage

```
mapGenes(mydata)
```

Arguments

mydata	Matrix or data.frame with RNA-seq gene TPM or microarray gene expression values for all samples to be deconvoluted, with gene symbols as row names and sample IDs as column names.

Value

The input matrix with updated gene names on the rows.

Examples

```
data(dataset_racle)
# mixture.fixgenes <- quantiseqr:::mapGenes(dataset_racle$expr_mat)
```

---

quantiplot *Plot the information on the tumor immune contexture*

Description

Plot the information on the tumor immune contexture, as extracted with `run_quantiseqr()`

Usage

```
quantiplot(obj)
```

Arguments

`obj`	An object, either • a `SummarizedExperiment` where the quantifications are stored • a simple data.frame object, as obtained by `run_quantiseqr()`

Value

A ggplot object

Examples

```
data(dataset_racle)
dim(dataset_racle$expr_mat)
res_quantiseq_run <- quantiseqr::run_quantiseq(
  expression_data = dataset_racle$expr_mat,
  signature_matrix = "TIL10",
  is_arraydata = FALSE,
  is_tumordata = TRUE,
  scale_mRNA = TRUE
)

# using a SummarizedExperiment object
library("SummarizedExperiment")
se_racle <- SummarizedExperiment(
  assays = List(
    abundance = dataset_racle$expr_mat
  ),
  colData = DataFrame(
    SampleName = colnames(dataset_racle$expr_mat)
  )
)

res_run_SE <- quantiseqr::run_quantiseq(
```

```
    expression_data = se_racle,
    signature_matrix = "TIL10",
    is_arraydata = FALSE,
    is_tumordata = TRUE,
    scale_mRNA = TRUE
)

quantiplot(res_quantiseq_run)
# equivalent to...
quantiplot(res_run_SE)
```

---

quanTIseq *Run quanTIseq deconvolution*

Description

Run quanTIseq deconvolution

Usage

```
quanTIseq(currsig, currmix, scaling = TRUE, method = "lsei")
```

Arguments

currsig	Signature matrix to be used for deconvolution (format: genes by cell types).
currmix	Mixture matrix to be deconvoluted (format: genes by samples).
scaling	Logical value. If set to FALSE, it disables the correction of cell-type-specific mRNA content bias. Default: TRUE
method	Character string, defining the deconvolution method to be used: `lsei` for constrained least squares regression, `hampel`, `huber`, or `bisquare` for robust regression with Huber, Hampel, or Tukey bisquare estimators, respectively. Default: `lsei`.

Value

A data.frame of cell fractions, cell types by samples.

Examples

```
data(dataset_racle)
mixture <- dataset_racle$expr_mat
signature.file <- system.file(
  "extdata", "TIL10_signature.txt", package = "quantiseqr", mustWork = TRUE)
signature <- read.table(signature.file, header = TRUE, sep = "\t", row.names = 1)
# cellfrac <- quantiseqr:::quanTIseq(mixture, signature)
```

quantiseqr-pkg *quantiseqr package*

Description

pkg description

Author(s)

Maintainer: Federico Marini <marinif@uni-mainz.de> (ORCID)

Authors:

- Francesca Finotello <francesca.finotello@i-med.ac.at> (ORCID)

quantiseq_helper *Helper functions for quanTIseq*

Description

Helper functions for quanTIseq

run_quantiseq *Run the quanTIseq algorithm*

Description

Use quanTIseq to deconvolute a gene expression matrix.

Usage

```
run_quantiseq(
  expression_data,
  signature_matrix = "TIL10",
  is_arraydata = FALSE,
  is_tumordata = FALSE,
  scale_mRNA = TRUE,
  method = "lsei",
  column = "gene_symbol",
  rm_genes = NULL,
  return_se = is(expression_data, "SummarizedExperiment")
)
```

Arguments

`expression_data`
: The gene expression information, containing the TPM values for the measured features. Can be provided as

  - a simple gene expression matrix, or a data frame (with HGNC gene symbols as row names and sample identifiers as column names)
  - an `ExpressionSet` object (from the Biobase package), where the HGNC gene symbols are provided in a column of the `fData` slot - that is specified by the `column` parameter below
  - a `SummarizedExperiment` object, or any of the derivative classes (e.g. DESeq2's `DESeqDataSet`), in which the assay (default: "abundance") is containing the TPMs as expected. Internally, `quantiseqr` handles the conversion to an object which is used in the deconvolution procedure.

`signature_matrix`
: Character string, specifying the name of the signature matrix. At the moment, only the original `TIL10` signature can be selected.

`is_arraydata`
: Logical value. Should be set to TRUE if the expression data are originating from microarray data. For RNA-seq data, this has to be FALSE (default value). If set to TRUE, the `rmgenes` parameter (see below) is set to "none".

`is_tumordata`
: Logical value. Should be set to TRUE if the expression data is from tumor samples. Default: FALSE (e.g. for RNA-seq from blood samples)

`scale_mRNA`
: Logical value. If set to FALSE, it disables the correction of cell-type-specific mRNA content bias. Default: TRUE

`method`
: Character string, defining the deconvolution method to be used: `lsei` for constrained least squares regression, `hampel`, `huber`, or `bisquare` for robust regression with Huber, Hampel, or Tukey bisquare estimators, respectively. Default: `lsei`.

`column`
: Character, specifies which column in the `fData` slot (for the ExpressionSet object) contains the information of the HGNC gene symbol identifiers

`rm_genes`
: Character vector, specifying which genes have to be excluded from the deconvolution analysis. It can be provided as

  - a vector of gene symbols (contained in the `expression_data`)
  - a single string among the choices of "none" (no genes are removed) and "default" (a list of genes with noisy expression RNA-seq data is removed, as explained in the quanTIseq paper). Default: "default" for RNA-seq data, "none" for microarrays.

`return_se`
: Logical value, controls the format of how the quantification is returned. If providing a `SummarizedExperiment`, it can simply extend its `colData` component, without the need to create a separate data frame as output.

Details

The values contained in the `expression_data` need to be provided as TPM values, as this is the format also used to store the `TIL10` signature, upon which quanTIseq builds to perform the immune cell type deconvolution. Expression data should *not* be provided in logarithmic scale.

If providing the `expression_data` as a `SummarizedExperiment`/`DESeqDataSet` object, it might be beneficial that this has been created via `tximport` - if this is the case, the assay named "abundance" will be automatically created upon importing the transcript quantification results.

Value

A data.frame containing the quantifications of the cell type proportions, or alternatively, if providing `expression_data` as `SummarizedExperiment` and setting `return_se` to TRUE, a `SummarizedExperiment` with the quantifications included by expanding the `colData` slot of the original object

References

F. Finotello, C. Mayer, C. Plattner, G. Laschober, D. Rieder, H. Hackl, A. Krogsdam, Z. Loncova, W. Posch, D. Wilflingseder, S. Sopper, M. Jsselsteijn, T. P. Brouwer, D. Johnsons, Y. Xu, Y. Wang, M. E. Sanders, M. V. Estrada, P. Ericsson-Gonzalez, P. Charoentong, J. Balko, N. F. d. C. C. de Miranda, Z. Trajanoski. "Molecular and pharmacological modulators of the tumor immune contexture revealed by deconvolution of RNA-seq data". Genome Medicine 2019;11(1):34. doi: 10.1186/s13073-019-0638-6.

C. Plattner, F. Finotello, D. Rieder. "Chapter Ten - Deconvoluting tumor-infiltrating immune cells from RNA-seq data using quanTIseq". Methods in Enzymology, 2020. doi: 10.1016/bs.mie.2019.05.056.

Examples

```
data(dataset_racle)
dim(dataset_racle$expr_mat)
res_quantiseq_run <- quantiseqr::run_quantiseq(
  expression_data = dataset_racle$expr_mat,
  signature_matrix = "TIL10",
  is_arraydata = FALSE,
  is_tumordata = TRUE,
  scale_mRNA = TRUE
)

# using a SummarizedExperiment object
library("SummarizedExperiment")
se_racle <- SummarizedExperiment(
  assays = List(
    abundance = dataset_racle$expr_mat
  ),
  colData = DataFrame(
    SampleName = colnames(dataset_racle$expr_mat)
  )
)

res_run_SE <- quantiseqr::run_quantiseq(
    expression_data = se_racle,
    signature_matrix = "TIL10",
    is_arraydata = FALSE,
    is_tumordata = TRUE,
    scale_mRNA = TRUE
)
```

se_to_matrix *SummarizedExperiment to matrix*

Description

SummarizedExperiment to matrix

Usage

```
se_to_matrix(se, assay = "abundance")
```

Arguments

se	A SummarizedExperiment object, or any of its derivates, which contains the information on the TPM expression values, which are stored in a specified assay slot.
assay	A character string, specifying the name of the assays component of the se object. Defaults to "abundance", as this is the common convention used e.g. by the tximport package to store the values imported from the transcript level quantifications

Value

A matrix object, containing the TPM values, ready to be used in the framework of quantiseqr

Examples

```
library("SummarizedExperiment")
library("macrophage")
data("gse", package = "macrophage")
se <- gse

# If using ENSEMBL or Gencode gene annotation, you might want to convert the row names
## in this case, the gene symbols are provided as rowData information
rownames(se) <- rowData(se)$SYMBOL

tpm_matrix <- se_to_matrix(se, assay = "abundance")

## otherwise, you can map the identifiers via
library("org.Hs.eg.db")
library("AnnotationDbi")
se <- gse
# keep the parts before the '.', used in the Gencode annotation
rownames(se) <- substr(rownames(se), 1, 15)
gene_names <- mapIds(org.Hs.eg.db,
                     keys = rownames(se),
                     column = "SYMBOL",
```

```
                      keytype = "ENSEMBL")
rownames(se) <- gene_names

# If you require to convert the counts to TPMs by hand, you need a vector of
# gene lengths as well, and then run this simple function on the count matrix
counts_to_tpm <- function(counts, lengths) {
  ratio <- counts / lengths
  mytpm <- ratio / sum(ratio) * 1e6
  return(mytpm)
}
# then run via
# tpmdata <- counts_to_tpm(count_matrix, genelength_vector)
```

`ti_quant_sim1700mixtures` *quanTIseq output for the simulation data of 1700 mixtures for RNA-seq data*

Description

quanTIseq output for the simulation data of 1700 mixtures for RNA-seq data

Details

quanTIseq output for the simulation data of 1700 mixtures for RNA-seq data, stored as a data.frame with 1700 rows (all the single instances of the different mixtures) as returned by `run_quantiseq()`. Column names, accordingly, contain the names of the component of the TIL10 signature, namely "B.cells", "Macrophages.M1", "Macrophages.M2", "Monocytes", "Neutrophils", "NK.cells", "T.cells.CD4", "T.cells.CD8", "Tregs", "Dendritic.cells", and "Other" (indicating for example a proxy for the amount of tumor tissue).

This can be compared (see Vignette for an example) to the ground truth information on the components of the mixtures.

References

Finotello, F., Mayer, C., Plattner, C. et al. Correction to: Molecular and pharmacological modulators of the tumor immune contexture revealed by deconvolution of RNA-seq data. Genome Med 11, 50 (2019). https://doi.org/10.1186/s13073-019-0655-5

Index

∗ **internal**
quantiseqr-pkg, 12

check_signature, 2

dataset_racle, 3
DClsei, 3
DCrr, 4

eset_to_matrix, 5
extract_ti_from_se, 6

fixMixture, 7

get_densities, 8

limSolve::lsei(), *4*

makeQN, 9
mapGenes, 9
MASS::rlm(), *5*

quantiplot, 10
quanTIseq, 11
quantiseq_helper, 12
quantiseqr *(*quantiseqr-pkg*)*, 12
quantiseqr-package *(*quantiseqr-pkg*)*, 12
quantiseqr-pkg, 12

run_quantiseq, 12

se_to_matrix, 15

ti_quant_sim1700mixtures, 16