Package 'tTEscanR'

November 20, 2026

Title A user-friendly integrative R package for quantifying and visualizing translation efficiency from sequencing data in diverse biological systems

Version 0.99.2

Description Translation elongation is dependent on codon-anticodon interactions, with suitable nucleotide pairing being essential for efficient translation. To quantify this relationship, we previously developed a computational pipeline (GitHub - wgao688/sc_tRNA_mRNA) that uses mRNA codon usage relative to tRNA anticodon availability as a proxy for theoretical translation efficiency (tTE). Here, we introduce tTEscanR, a powerful and user-friendly R-based package that extends this approach to quantify translation efficiency from both bulk and single-cell sequencing data. tTEscanR is a versatile tool for exploring translation efficiency in diverse cellular processes, disease mechanisms, and therapeutic development. It also features an advanced visualization module to generate high-quality plots, enhancing result interpretation and communication.

License MIT + file LICENSE

Imports stats, SummarizedExperiment, Biostrings, S4Vectors, MultiAssayExperiment, Matrix, DESeq2, matrixStats, tibble, dplyr, tRNAscanImport, GenomicRanges, Signac, utils, biomaRt, stringr, ggplot2, ggrepel, uwot, Rtsne, dendsort, grid, ComplexHeatmap, viridis, tidyr, ggpubr, patchwork, purrr, magrittr, rlang

Encoding UTF-8

Depends R (>= 4.6.0)

VignetteBuilder knitr

URL https://github.com/avarassanchez/tTEscanR

BugReports https://github.com/avarassanchez/tTEscanR/issues

Suggests testthat, withr, mockery, knitr, rmarkdown, BiocStyle, devtools, pheatmap, Seurat, IRanges, GenomeInfoDb, Rsamtools

Config/testthat/edition 3

biocViews Software, Epitranscriptomics, Transcriptomics, GeneExpression, GeneRegulation, Sequencing, SingleCell, CellBasedAssays

Roxygen list(markdown = TRUE, r6 = FALSE)

Config/roxygen2/version 8.1.1

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

git_branch devel

git_last_commit 81c5457

git_last_commit_date 2026-10-16

Repository Bioconductor 3.24

Date/Publication 2026-11-19

Author Ana Varas-Sánchez [aut, cre] (ORCID:
<https://orcid.org/0009-0006-0187-8968>),
Carlos Gallardo-Dodd [aut] (ORCID:
<https://orcid.org/0000-0002-6086-0550>),
William Gao [aut] (ORCID: <https://orcid.org/0000-0002-9415-331X>),
Claudia Kutter [ths] (ORCID: <https://orcid.org/0000-0002-8047-0058>),
Karolinska Institutet [fnd] (grant_id: KID-2021-00308, KID-2023-01298),
Swedish Research Council [fnd] (grant_id: 2023-02780),
Cancerfonden [fnd] (grant_id: 22 2246 Pj, 25 4703 Pj)

Maintainer Ana Varas-Sánchez <ana.varas@scilifelab.se>

Contents

tTEscanR-package . . . 3
computeAAUsage . . . 4
computeAnticodonUsage . . . 5
computeCodonUsage . . . 5
computeCorrelationBackground . . . 7
computeDEResults . . . 8
computeExonicBackground . . . 9
computeMeanUsage . . . 10
computeTheoreticalTE . . . 11
controlVerbose . . . 12
createObject . . . 13
default_tTEscanR_metadata . . . 14
default_tTEscanR_mRNA_data . . . 14
default_tTEscanR_tRNA_data . . . 15
extractCodons . . . 16
featuresToAA . . . 16
getCodonFreq . . . 17
getPermutationDist . . . 19
groupConditions . . . 20
mergeMatrices . . . 20
obtainSignificance . . . 21
plotCorrelation . . . 22
plotDEResults . . . 24
plotDistribution . . . 25
plotPermutation . . . 27
plotProportion . . . 29
plotTargetComparison . . . 31
plotTEscore . . . 32
runDEAnalysis . . . 34
runPipeline . . . 35
showPoolContribution . . . 37
transformFormat . . . 38

tRNAFilterCuts 39
tRNAGetMatrix 40
tRNASetCutoff 41
tRNASetGenes 42
updateObject 42

Index **44**

`tTEscanR-package` *tTEscanR: A user-friendly integrative R package for quantifying and visualizing translation efficiency from sequencing data in diverse biological systems*

Description

Translation elongation is dependent on codon-anticodon interactions, with suitable nucleotide pairing being essential for efficient translation. To quantify this relationship, we previously developed a computational pipeline (GitHub - wgao688/sc_tRNA_mRNA) that uses mRNA codon usage relative to tRNA anticodon availability as a proxy for theoretical translation efficiency (tTE). Here, we introduce tTEscanR, a powerful and user-friendly R-based package that extends this approach to quantify translation efficiency from both bulk and single-cell sequencing data. tTEscanR is a versatile tool for exploring translation efficiency in diverse cellular processes, disease mechanisms, and therapeutic development. It also features an advanced visualization module to generate high-quality plots, enhancing result interpretation and communication.

Author(s)

Maintainer: Ana Varas-Sánchez <ana.varas@scilifelab.se> (ORCID)

Authors:

- Ana Varas-Sánchez <ana.varas@scilifelab.se> (ORCID)
- Carlos Gallardo-Dodd <carlos.gallardo@scilifelab.se> (ORCID)
- William Gao <william.gao@pennmedicine.upenn.edu> (ORCID)

Other contributors:

- Claudia Kutter <claudia.kutter@ki.se> (ORCID) [thesis advisor]
- Karolinska Institutet (grant_id: KID-2021-00308, KID-2023-01298) [funder]
- Swedish Research Council (grant_id: 2023-02780) [funder]
- Cancerfonden (grant_id: 22 2246 Pj, 25 4703 Pj) [funder]

See Also

Useful links:

- https://github.com/avarassanchez/tTEscanR
- Report bugs at https://github.com/avarassanchez/tTEscanR/issues

computeAAUsage *Compute Amino Acid (AA) Demand or Supply*

Description

Calculates the amino acid (AA) demand and/or supply from a codon and/or anticodon usage matrices of a mRNA expression and/or tRNA abundance count matrix. It aggregates the contribution of their features based on the standard genetic code (mapping codons/anticodons to AA). The resulting values reflect total usage (demand) or availability (supply) of each amino acid, depending on the input type.

Usage

```
computeAAUsage(object, level, genetic_code = "Standard", overwrite = FALSE)
```

Arguments

`object`	A `MultiAssayExperiment` containing codon and/or anticodon usage assays to be analyzed.
`level`	Either `"demand"`, `"supply"` or `"both"` to indicate which analysis to perform.
`genetic_code`	A `character` string to specify the genetic code to be used. Defaults to `"Standard"`.
`overwrite`	Logical; if `TRUE`, overwrites any existing assay in the `object`. Defaults to `FALSE`

Details

In order to generalize the analysis to any organism available in Ensembl, tTEscanR can be used with the 33 different genetic codes described by NCBI Taxonomy:

“Standard”, "Vertebrate Mitochondrial”, "Yeast Mitochondrial”, ”Mold Mitochondrial; Protozoan Mitochondrial; Coelenterate Mitochondrial; Mycoplasma; Spiroplasma”, "Invertebrate Mitochondrial”, "Ciliate Nuclear; Dasycladacean Nuclear; Hexamita Nuclear”, "Echinoderm Mitochondrial; Flatworm Mitochondrial”, ”Euplotid Nuclear”, "Bacterial, Archaeal and Plant Plastid”, "Alternative Yeast Nuclear”, "Ascidian Mitochondrial”, "Alternative Flatworm Mitochondrial”, "Blepharisma Macronuclear”, "Chlorophycean Mitochondrial”, "Trematode Mitochondrial”, "Scenedesmus obliquus mitochondrial”, "Thraustochytrium mitochondrial code”, "Rhabdopleuridae Mitochondrial”, "Candidate Division SR1 and Gracilibacteria”, "Pachysolen tannophilus Nuclear”, "Karyorelict Nuclear”, "Condylostoma Nuclear”, "Mesodinium Nuclear”, "Peritrich Nuclear”, "Blastocrithidia Nuclear”, "Balanophoraceae Plastid”, "Cephalodiscidae Mitochondrial"

Value

An updated `MultiAssayExperiment` containing a new layer of information in the `assays` representing the `"AADemand"` and/or `"AASupply"`.

Examples

```
data("default_tTEscanR_tRNA_data", package = "tTEscanR")

tTEscanR_obj <- createObject(
    counts = default_tTEscanR_tRNA_data,
    assay = "tRNA"
)
```

```
tTEscanR_obj <- computeAnticodonUsage(object = tTEscanR_obj)
tTEscanR_obj <- computeAAUsage(object = tTEscanR_obj, level = "supply")
```

`computeAnticodonUsage` *Compute Anticodon Usage from tRNA Gene Expression Data*

Description

Calculates **anticodon usage profiles** from tRNA gene abundance data. It summarizes the abundance of tRNAs by their anticodon identity, which can be used to estimate the tRNA supply landscape. The tRNA gene names need to be properly annotated for proper recognition.

Usage

```
computeAnticodonUsage(object, overwrite = FALSE)
```

Arguments

`object`	A `MultiAssayExperiment` containing a tRNA assay.
`overwrite`	Logical; if `TRUE`, overwrites any existing assay and metadata in the `object`. Defaults to `FALSE`.

Value

An updated `MultiAssayExperiment` containing a new layer of information in the `assays` representing the `"AnticodonUsage"`.

Examples

```
data("default_tTEscanR_tRNA_data", package = "tTEscanR")

tTEscanR_obj <- createObject(
    counts = default_tTEscanR_tRNA_data,
    assay = "tRNA"
)
tTEscanR_obj <- computeAnticodonUsage(object = tTEscanR_obj)
```

`computeCodonUsage` *Compute Codon Usage from mRNA Gene Expression Data*

Description

Estimates **codon usage profiles** based on gene-level mRNA expression data. It optionally accepts pre-computed codon frequency tables or uses internally generated default tables when not provided. When enabled, it can evaluate the correlation between background codon composition and observed mean codon usage.

Usage

```
computeCodonUsage(
  object,
  codon_freq = NULL,
  species = NULL,
  additional_metrics = TRUE,
  reduce = 100,
  corr_method = "spearman",
  overwrite = FALSE
)
```

Arguments

object	A MultiAssayExperiment containing a mRNA assay.
codon_freq	Optional; a user-provided codon frequency-per-gene table. If necessary, it can be computed using getCodonFreq.
species	Optional; either "hg38" (human) or "mm39" (mouse) to load the default settings. Required if codon_freq is not provided.
additional_metrics	Logical; if TRUE, computes: (i) codon exonic background, (ii) mean codon usage, and (iii) correlation between the previous metrics. Defaults to TRUE.
reduce	Numeric; a scaling factor used to normalize large expression values that exceed R's handling capacity. Defaults to 100.
corr_method	A correlation method accepted by cor. Required if additional_metrics is TRUE. Defaults to "spearman".
overwrite	Logical; if TRUE, overwrites any existing assay and metadata in the object. Defaults to FALSE.

Value

An updated MultiAssayExperiment containing a new layer of information in the assays representing the "CodonUsage". Additional computations will be stored in the param slot as "CodonUsage_AdditionalMetri

Examples

```
data("default_tTEscanR_mRNA_data", package = "tTEscanR")
data("default_tTEscanR_metadata", package = "tTEscanR")

tTEscanR_obj <- createObject(
    counts = default_tTEscanR_mRNA_data,
    assay = "mRNA",
    meta_data = default_tTEscanR_metadata,
    sample_id = "conditions",
    params = list("CorrectionFactor" = "tissue")
)
tTEscanR_obj <- computeCodonUsage(
    object = tTEscanR_obj, species = "hg38",
    additional_metrics = FALSE, reduce = 1000
)
```

computeCorrelationBackground *Compute the Correlation Between Mean Usage and Exonic Background*

Description

This function calculates the **correlation** between observed **mean usage** and the **exonic background**. It provides a metric for evaluating how much usage is driven by underlying sequence composition versus condition-specific expression.

Usage

```
computeCorrelationBackground(mean, background, corr_method = "spearman")
```

Arguments

mean	A matrix containing the mean usage across conditions. Can be computed using computeMeanUsage.
background	A matrix or table containing the frequencies in exonic regions. Can be computed using computeExonicBackground.
corr_method	A correlation method accepted by cor. Defaults to "spearman".

Value

Integer; correlation information between mean and background.

Examples

```
data("default_tTEscanR_mRNA_data", package = "tTEscanR")
data("default_tTEscanR_metadata", package = "tTEscanR")

tTEscanR_obj <- createObject(
    counts = default_tTEscanR_mRNA_data,
    assay = "mRNA"
)
tTEscanR_obj <- computeCodonUsage(
    object = tTEscanR_obj, species = "hg38",
    additional_metrics = FALSE, reduce = 1000
)
codon_usage <- SummarizedExperiment::assay(tTEscanR_obj, "CodonUsage")
exonic_background <- computeExonicBackground(data = codon_usage)
# Input: expression count matrix, need to provide metadata & batch parameters
mean_codon_usage <- computeMeanUsage(
    data = codon_usage,
    mode = "raw",
    metadata = default_tTEscanR_metadata,
    batch = "tissue"
)
corr_back <- computeCorrelationBackground(
    mean = mean_codon_usage,
    background = exonic_background
)
```

`computeDEResults` *Compute the DESeq2 Analysis*

Description

Compute the DESeq2 Analysis

Usage

```
computeDEResults(
  list_data,
  metadata,
  target = NULL,
  batch = NULL,
  reference = NULL,
  reduce = 100,
  padj_threshold = 0.05,
  compute_pairwise = TRUE
)
```

Arguments

`list_data`	A `list` of matrices with features (i.e. genes, codons, anticodons or amino acids) as rows and samples or conditions as columns. The list can be named or unnamed.
`metadata`	A `data.frame` with the data associated to the conditions in the matrices of `list_data`. There has to be one column with the same labels as the column names.
`target`	Optional; a factor based on `metadata` columns to define the comparisons to perform in a targeted analysis. Defaults to `NULL`.
`batch`	Optional; name of the categorical variable in `metadata` to correct the data. Required if `dim_reduct` specified.
`reference`	Optional; factor from the `batch` variable to use as reference for the corrections. If not specified, the 1st factor that appears will be used instead.
`reduce`	Numeric; a scaling factor used to normalize large expression values that exceed R's handling capacity. Defaults to 100.
`padj_threshold`	Numeric; p-value threshold used for highlighting significant features in the volcano plot. Defaults to 0.05.
`compute_pairwise`	Logical; if `TRUE`, computes all the pairwise comparisons based on the conditions included in the input data.

Value

A DESeq2 object with the normalized and vst counts.

Examples

```
data("default_tTEscanR_tRNA_data", package = "tTEscanR")
data("default_tTEscanR_metadata", package = "tTEscanR")

DE_analysis <- computeDEResults(
    list_data = list(tRNA = default_tTEscanR_tRNA_data),
    metadata = default_tTEscanR_metadata, batch = "tissue",
    compute_pairwise = FALSE
)
```

`computeExonicBackground` *Compute the Exonic Background of the Codon/Anticodon Usage*

Description

This function calculates the **codon/anticodon usage background** based solely on exonic sequence composition, independent of expression levels. It provides a reference distribution of codon/anticodon frequencies across conditions, used to normalize/compare against observed usage patterns derived from expression data.

Usage

```
computeExonicBackground(data)
```

Arguments

`data`	A codon usage matrix with codons as rows and conditions or samples as columns.

Value

A `matrix` with the codon/anticodon background.

Examples

```
data("default_tTEscanR_mRNA_data", package = "tTEscanR")

tTEscanR_obj <- createObject(
    counts = default_tTEscanR_mRNA_data,
    assay = "mRNA"
)
tTEscanR_obj <- computeCodonUsage(
    object = tTEscanR_obj, species = "hg38",
    additional_metrics = FALSE, reduce = 1000
)
codon_usage <- SummarizedExperiment::assay(tTEscanR_obj, "CodonUsage")
exonic_background <- computeExonicBackground(data = codon_usage)
```

computeMeanUsage *Compute Usage Across Conditions (Mean Usage)*

Description

This function computes the **average usage** of codons, anticodons, or amino acids across conditions, useful for summarizing feature usage trends across sample groups. It supports direct input of a count matrix and extraction from a `tTEscanR_Object`. When the input `data` is a `tTEscanR_Object` the parameters `metadata` and `batch` will be extracted from the object, and ignored if specified as input parameters. Therefore, variables `assay` and `metadata` need to be coherent with the rules described in `createObject`.

Usage

```
computeMeanUsage(
  data,
  assay = NULL,
  metadata = NULL,
  batch = NULL,
  mode = c("raw", "size-corrected", "long_format")
)
```

Arguments

`data`	A `tTEscanR_Object` or expression count `matrix` (with codons, anticodons or amino acids as features).
`assay`	Optional; a character string specifying the name of the assay to retrieve from the `tTEscanR_Object`.
`metadata`	Optional; a `data.frame` with the meta-information related with the conditions in `data`. There has to be one column with the same labels as the column names.
`batch`	Optional; a factor based on `metadata` columns to define the variable to correct for. Required if `mode` is `"raw"` and `data` is not a `tTEscanR_Object`.
`mode`	Either `"raw"`, `"size-corrected"` or `"long-format"` to specify the format that the input data belongs to. Defaults to `"raw"`.

Value

An updated `tTEscanR_Object` if `data` is a `tTEscanR_Object`. A `data.frame` containing a new layer of information representing the mean codon usage if `data` is an expression count matrix.

Examples

```
data("default_tTEscanR_tRNA_data", package = "tTEscanR")
data("default_tTEscanR_metadata", package = "tTEscanR")

tTEscanR_obj <- createObject(
    counts = default_tTEscanR_tRNA_data,
    assay = "tRNA",
    meta_data = default_tTEscanR_metadata,
    sample_id = "conditions",
    params = list("CorrectionFactor" = "tissue")
```

```
)
# Input: tTEscanR object containing metadata and batch parameters
tTEscanR_obj <- computeAnticodonUsage(object = tTEscanR_obj)
anticodon_mean_usage <- computeMeanUsage(
    data = tTEscanR_obj,
    assay = "AnticodonUsage"
)
```

`computeTheoreticalTE` *Compute the Theoretical Translation Efficiency (tTE) score*

Description

This function calculates the theoretical translation efficiency (tTE) score by integrating codon-anticodon usage and/or amino acid demand-supply across conditions.

Usage

```
computeTheoreticalTE(
  object,
  level = c("codon", "aa", "both"),
  genetic_code = "Standard",
  compute_significance = TRUE,
  overwrite = FALSE,
  corr_method = c("spearman", "pearson", "kendall")
)
```

Arguments

`object`	A `MultiAssayExperiment` containing a mRNA and a codon usage assay and/or a tRNA and an anticodon usage assay.
`level`	Either `"codon"`, `"aa"` or `"both"` to indicate which analysis to perform.
`genetic_code`	A `character` string to specify the genetic code to be used. Defaults to `"Standard"`.
`compute_significance`	Logical; if `TRUE`, computes the statistical significance (p-value) of the tTE scores. Defaults to `TRUE`.
`overwrite`	Logical; if `TRUE`, overwrites any existing tTE table in the `object`. Defaults to `FALSE`.
`corr_method`	A correlation method accepted by `cor`. Defaults to `"spearman"`.

Value

An updated `MultiAssayExperiment` containing a new layer of information representing the translation efficiency table for the matching conditions in the mRNA and tRNA data.

Examples

```
data("default_tTEscanR_mRNA_data", package = "tTEscanR")
data("default_tTEscanR_tRNA_data", package = "tTEscanR")
data("default_tTEscanR_metadata", package = "tTEscanR")

tTEscanR_obj <- createObject(
    counts = list(
        mRNA = default_tTEscanR_mRNA_data, tRNA = default_tTEscanR_tRNA_data
    ),
    meta_data = default_tTEscanR_metadata, sample_id = "conditions",
    params = list("CorrectionFactor" = "tissue")
)
tTEscanR_obj <- computeCodonUsage(
    object = tTEscanR_obj, species = "hg38",
    additional_metrics = FALSE, reduce = 10000
)
tTEscanR_obj <- computeAnticodonUsage(object = tTEscanR_obj)
tTEscanR_obj <- computeTheoreticalTE(
    object = tTEscanR_obj, level = "codon", compute_significance = FALSE
)
```

`controlVerbose` *Control tTEscanR message output*

Description

Control tTEscanR message output

Usage

```
controlVerbose(quiet = TRUE)
```

Arguments

`quiet`	Logical; if `TRUE`, suppresses informative messages.

Value

Invisible; quiet state

Examples

```
controlVerbose(TRUE) # Silence package messages
controlVerbose(FALSE) # Enable package messages
```

createObject *Create a tTEscanR object as a MultiAssayExperiment*

Description

Initializes a MultiAssayExperiment object to store translation efficiency data layers.

Usage

```
createObject(
  counts,
  assay = "mRNA",
  meta_data = NULL,
  sample_id = NULL,
  params = list()
)
```

Arguments

counts	A matrix, data.frame or list of count matrices.
assay	Optional; a character string or vector specifying the assay name(s) when counts is a matrix, data.frame or unnamed list. Defaults to "mRNA".
meta_data	Optional; a data.frame of sample metadata.
sample_id	Optional; a character string naming the column in meta.data to use as sample row names.
params	Optional; a list of global run parameters.

Value

A MultiAssayExperiment object.

Examples

```
data("default_tTEscanR_mRNA_data", package = "tTEscanR")
data("default_tTEscanR_tRNA_data", package = "tTEscanR")
data("default_tTEscanR_metadata", package = "tTEscanR")

tTEscanR_obj <- createObject(
    counts = list(
        mRNA = default_tTEscanR_mRNA_data,
        tRNA = default_tTEscanR_tRNA_data
    ),
    meta_data = default_tTEscanR_metadata,
    sample_id = "conditions"
)
```

`default_tTEscanR_metadata`

Metadata of mRNA and tRNA data

Description

Metadata table generated from the cell-types included in `"default_tTEscanR_mRNA_data"` and `"default_tTEscanR_tRNA_data"` used in package vignettes and examples.

Usage

```
data(default_tTEscanR_metadata)
```

Format

Manually data.frame generated to define experimental variables for the sample column identifiers present in the expression datasets derived from Gao et al. (2022), Cao et al. (2020), and Domcke et al. (2020). The columns included are:

tissue The sequencing batch or sample origin

cell.type The experimental group or cell type

conditions The combination of the previous items as will be referred in the columns of the count matrices

References

Gao W, Gallardo-Dodd CJ, Kutter C (2022). "Cell type-specific analysis by single-cell profiling identifies a stable mammalian tRNA-mRNA interface and increased translation efficiency in neurons." *Genome Research*, 32(1), 97–110. doi:10.1101/gr.275944.121

Cao J, O'Day DR, Pliner HA, et al. (2020). "A human cell atlas of fetal gene expression." *Science*, 370(6518), eaba7721. doi:10.1126/science.aba7721

Domcke S, Hill AJ, Daza RM, et al. (2020). "A human cell atlas of fetal chromatin accessibility." *Science*, 370(6518), eaba7612. doi:10.1126/science.aba7612

`default_tTEscanR_mRNA_data`

mRNA Expression Data Subset

Description

A subset of preprocessed mRNA gene expression count data used in package vignettes and examples.

Usage

```
data(default_tTEscanR_mRNA_data)
```

Format

A matrix where rows represent genes and columns represent individual samples or cell-types.

Source

Subset of preprocessed count table generated by Gao et al. (2022), derived from raw sequencing data originally published by Cao et al. (2020).

References

Gao W, Gallardo-Dodd CJ, Kutter C (2022). "Cell type-specific analysis by single-cell profiling identifies a stable mammalian tRNA-mRNA interface and increased translation efficiency in neurons." *Genome Research*, 32(1), 97–110. doi:10.1101/gr.275944.121

Cao J, O'Day DR, Pliner HA, et al. (2020). "A human cell atlas of fetal gene expression." *Science*, 370(6518), eaba7721. doi:10.1126/science.aba7721

`default_tTEscanR_tRNA_data`

tRNA Expression Data Subset

Description

A subset of preprocessed tRNA gene abindace count data used in package vignettes and examples.

Usage

```
data(default_tTEscanR_tRNA_data)
```

Format

A matrix or data frame where rows represent tRNA gene names and columns represent individual samples or cell-types.

Source

Subset of preprocessed count table generated by Gao et al. (2022), derived from raw sequencing data originally published by Domcke et al. (2020).

References

Gao W, Gallardo-Dodd CJ, Kutter C (2022). "Cell type-specific analysis by single-cell profiling identifies a stable mammalian tRNA-mRNA interface and increased translation efficiency in neurons." *Genome Research*, 32(1), 97–110. doi:10.1101/gr.275944.121

Domcke S, Hill AJ, Daza RM, et al. (2020). "A human cell atlas of fetal chromatin accessibility." *Science*, 370(6518), eaba7612. doi:10.1126/science.aba7612

extractCodons *Extract Codon Composition of Sequences*

Description

This function analyzes a given set of nucleotide sequences and computes the count of each codon present.

Usage

```
extractCodons(sequences)
```

Arguments

sequences	A list of nucleotide sequences (character strings) from which to extract the codon composition.

Value

Codon frequency per gene table of the sequences.

Examples

```
codon_composition <- extractCodons(sequences = list(
    "ATGCGTACG",
    "TTAAGGCCG"
))
```

featuresToAA *Relates Codons, Anticodons and their corresponding Amino Acids*

Description

This function converts: codons and anticodons into anticodons, codons or amino acids based on the genetic code.

Usage

```
featuresToAA(
  data,
  position = NULL,
  genetic_code = "Standard",
  notation_from = c("codon", "anticodon"),
  notation_to = c("aa", "anticodon", "codon")
)
```

Arguments

data	A character vector or data.frame containing gene names or features to be translated.
position	Optional; either "row", "column" or <column_name> to specify the location of the genes or features in data. Required if data_to_transalte is a data.frame.
genetic_code	A character string to specify the genetic code to be used. Defaults to "Standard".
notation_from	Either "codon" or "anticodon" to select the input format of the features in data_to_translate.
notation_to	Either "codon", "anticodon" or "aa" to select the output format of the features in data.

Value

Translated features (codons, anticodons or amino acids) from data_to_translate.

Examples

```
data(default_tTEscanR_mRNA_data, default_tTEscanR_metadata)
tTEscanR_obj <- createObject(
    counts = default_tTEscanR_mRNA_data,
    assay = "mRNA",
    meta_data = default_tTEscanR_metadata,
    sample_id = "conditions",
    params = list("CorrectionFactor" = "tissue")
)
tTEscanR_obj <- computeCodonUsage(
    object = tTEscanR_obj, codon_freq = NULL,
    species = "hg38", additional_metrics = FALSE
)
codons <- rownames(SummarizedExperiment::assay(tTEscanR_obj, "CodonUsage"))
codons_to_AA <- featuresToAA(
    data = codons, notation_from = "codon", notation_to = "aa"
)
codons_to_anticodons <- featuresToAA(
    data = codons,
    notation_from = "codon", notation_to = "anticodon"
)
```

---

getCodonFreq *Compute Codon Frequency-per-Gene Table*

Description

This function computes codon usage frequencies for each gene based on a provided set of gene sequences. It can optionally subset the analysis to specific transcripts and apply filtering criteria when multiple transcripts are available for the same gene. Consequently, no filter parameter will be considered if parameters transcripts and genes_file are given.

Usage

```
getCodonFreq(
  dataset_name = NULL,
  genes_file = NULL,
  transcripts = NULL,
  retain_mitochondrial = FALSE,
  retain_unannotated = FALSE,
  retain_geneversion = TRUE,
  filter = c("canonical", "length"),
 out_format = c("external_gene_name", "ensembl_transcript_id", "ensembl_gene_id")
)
```

Arguments

dataset_name	A character string specifying the Ensembl species dataset name (e.g. "hsapiens_gene_ensembl").
genes_file	Optional; a path to a FASTA file.
transcripts	Optional; a character vector of transcripts or gene IDs to subset the analysis.
retain_mitochondrial	Logical; if FALSE filters out the mitochondrial genes. Defaults to FALSE.
retain_unannotated	Logical; if FALSE filters out the gene names that do not have an "external_gene_name" identifier. Defaults to FALSE.
retain_geneversion	Logical; if FALSE retains the gene versions from the "ensembl_gene_id" identifier. Defaults to TRUE.
filter	Either "canonical" or "length" (longest transcript) to specify which transcript to choose if several are available for the same gene.
out_format	Either "external_gene_name", "ensembl_gene_id" or "ensembl_transcript_id" to specify annotation to use in the output codon frequency-per-gene table.

Value

Codon frequency-per-gene table and a translator gene annotation table (if available).

Examples

```
tryCatch({
    codon_freq_results_canonical_hg38 <- getCodonFreq(
        dataset_name = "hsapiens_gene_ensembl", filter = "canonical",
        retain_geneversion = TRUE, retain_mitochondrial = FALSE,
        out_format = "external_gene_name"
    )
}, error = function(i) {
    message("Unable to connect to Ensembl: ", i$message)
})
```

`getPermutationDist` *Run Codon Usage Permutation Test (Background)*

Description

This function performs a **permutation test** to compare a mRNA dataset to the reference codon frequency-per-gene matrix.

Usage

```
getPermutationDist(
  n_permut = 1000,
  n_features = 100,
  target_data = NULL,
  codon_freq = NULL,
  species = NULL
)
```

Arguments

`n_permut`	Numeric; number of permutations to perform. Defaults to 1000.
`n_features`	Numeric; number of features to select in each permutation. Defaults to 100. If `target_data` is given this parameter will take its length.
`target_data`	Optional; a mRNA expression count matrix with features as rows and conditions as columns.
`codon_freq`	Optional; a user-provided codon frequency per gene table. If necessary, it can be computed using `getCodonFreq`.
`species`	Optional, a `character` string specifying the species reference genome version (used if `codon_freq` is not provided or `translate` is `TRUE`). Supported values include `"hg38"` (human) and `"mm39"` (mouse).

Value

A table with the codons and their frequencies after computing all the permutations.

Examples

```
data("default_tTEscanR_mRNA_data", package = "tTEscanR")

genes <- default_tTEscanR_mRNA_data[1:20, ]
permut <- getPermutationDist(
    n_permut = 100, target_data = genes,
    species = "hg38"
)
```

groupConditions *Aggregates data by group*

Description

This function calculates the row sums of a given matrix to combine columns that share the same group.

Usage

```
groupConditions(data, group_labels)
```

Arguments

data	A matrix with the features to group for as columns.
group_labels	A vector with the metadata features to group the columns in data

Value

A matrix with the conditions merged based on the metadata.

Examples

```
data <- data.frame(
    sample_1 = c(10, 5, 20), sample_2 = c(15, 8, 25),
    sample_3 = c(12, 6, 22), sample_4 = c(1, 2, 3),
    sample_5 = c(4, 5, 6), sample_6 = c(7, 8, 9)
)
rownames(data) <- c("gene_1", "gene_2", "gene_3")
groups <- c("cond_A", "cond_A", "cond_A", "cond_B", "cond_B", "cond_B")
data_combined <- groupConditions(data = data, group_labels = groups)
```

mergeMatrices *Combine large matrices*

Description

This function efficiently combines individual matrices.

Usage

```
mergeMatrices(...)
```

Arguments

...	A variable number of matrix.

Value

A single sparse matrix with as a combination of all the input matrices.

Examples

```
df1 <- matrix(c(1, 0, 0, 2), nrow = 2, dimnames = list(
    c("geneA", "geneB"),
    c("s1", "s2")
))
df2 <- matrix(c(3, 0, 0, 4), nrow = 2, dimnames = list(
    c("geneB", "geneC"),
    c("s2", "s3")
))
merged_matrix <- mergeMatrices(df1, df2)
```

---

obtainSignificance *Assess Significance (P-value) & Corrects for Multiple Hypothesis Testing*

Description

Assess Significance (P-value) & Corrects for Multiple Hypothesis Testing

Usage

```
obtainSignificance(dist, value, padj_threshold = 0.05)
```

Arguments

`dist`	A table with the codons and their frequencies after completing a permutation test. Output from `getPermutationDist`.
`value`	A `list` of `data.frame` of the codon exonic background of a mRNA gene expression matrix. The codon exonic background can be computed in `computeCodonUsage` using the additional_metrics parameter or directly running `computeExonicBackground`.
`padj_threshold`	Numeric; p-value threshold used for highlighting significant features in the volcano plot. Defaults to 0.05.

Value

A table with the codon exonic background and their significance level before (p-value) and after the correction (p-adjusted value).

Examples

```
data("default_tTEscanR_mRNA_data", package = "tTEscanR")
data("default_tTEscanR_metadata", package = "tTEscanR")

selected_genes <- default_tTEscanR_mRNA_data[1:20, ]
permutation_test <- getPermutationDist(
    n_permut = 100, target_data = selected_genes, species = "hg38"
)
tTEscanR_obj <- createObject(
    counts = default_tTEscanR_mRNA_data,
    assay = "mRNA",
    meta_data = default_tTEscanR_metadata,
    sample_id = "conditions",
```

```
    params = list("CorrectionFactor" = "tissue")
)
tTEscanR_obj <- computeCodonUsage(
    object = tTEscanR_obj, species = "hg38",
    additional_metrics = FALSE, reduce = 1000
)

codon_usage <- SummarizedExperiment::assay(tTEscanR_obj, "CodonUsage")
codon_background <- rowSums(codon_usage) / sum(rowSums(codon_usage))
codons_to_AA <- featuresToAA(
    data = names(codon_background),
    notation_from = "codon", notation_to = "aa"
)
codon_background <- data.frame(
    group = codons_to_AA, codon = names(codon_background),
    freq = as.numeric(codon_background), row.names = NULL
)
significance <- obtainSignificance(
    dist = permutation_test, value = codon_background
)
```

`plotCorrelation` *Correlation Plot: Exonic Codon Background - Mean Codon Usage*

Description

This function generates a visualization to correlate different parameters of the `data`. The input data (`data`) is expected to be in long format and contain a minimum information regarding the features, the conditions they belong to and their usage counts. A common usage of this kind of plot is to represent the codon frequencies correlations of the exonic background and the mean codon usage across conditions. Based on the selected `plot` parameter the user can select the most convenient layout to display the data. It is crucial to ensure consistency between the name of the columns in `data` and the parameters `x_axis_col`, `y_axis_col` and `condition_col`.

Usage

```
plotCorrelation(
  data,
  plot = "MeanCodonUsage",
  x_axis_col,
  y_axis_col,
  condition_col,
  extra_val = NULL,
  label_col = NULL,
  color_palette = NULL,
  filename = NULL,
  show_legend = "none",
  targeted_arg = NULL,
  add_titles = TRUE
)
```

Arguments

data	A long format table. This format can be obtained using transformFormat.
plot	Either "MeanCodonUsage" (default) or "PoolDiversity" to indicate the data source.
x_axis_col	Name of the categorical variable to reflect in the plot.
y_axis_col	Name of the numerical variable to reflect in the plot.
condition_col	Name of the categorical variable to group the data points.
extra_val	Optional; variable with additional information to include in the plot (e.g. correlation value).
label_col	Name of the categorical variable to label the data points.
color_palette	Optional; a vector of color codes to customize plot appearance.
filename	Optional; name for the plot to be saved.
show_legend	Either "none" (default), "top", "bottom", "right" or "left" to specify the position of the legend.
targeted_arg	Optional; a vector defining key feature clusters to highlight or label.
add_titles	Logical; if TRUE, includes titles in the plot. Defaults to TRUE.

Value

A ggplot object representing the correlation. If save_format is provided, the plot will also be saved to the specified location.

Examples

```
data("default_tTEscanR_mRNA_data", package = "tTEscanR")
data("default_tTEscanR_metadata", package = "tTEscanR")

# Define the object and compute the codon usage
tTEscanR_obj <- createObject(
    counts = default_tTEscanR_mRNA_data, assay = "mRNA",
    meta_data = default_tTEscanR_metadata,
    sample_id = "conditions",
    params = list("CorrectionFactor" = "tissue")
)
tTEscanR_obj <- computeCodonUsage(
    object = tTEscanR_obj, species = "hg38",
    additional_metrics = TRUE, reduce = 1000
)

# Compute and extract the mean codon usage
meta <- S4Vectors::metadata(tTEscanR_obj)
additional_metrics <- meta[["CodonUsage_AdditionalMetrics"]]
mean_codon_usage <- additional_metrics$MeanCodonUsage
exonic_background <- additional_metrics$CodonExonicBackground
exonic_background <- as.data.frame(exonic_background)
correlation_mean_background <- cbind(mean_codon_usage, exonic_background)

plotCorrelation(
    data = correlation_mean_background, plot = "MeanCodonUsage",
    x_axis_col = "mean_usage_across_conditions",
    y_axis_col = "exonic_background", condition_col = "feature",
```

```
    extra_val = additional_metrics$MeanCodonCorr, add_titles = TRUE,
    show_legend = "none"
)
```

---

plotDEResults *Generates Visualizations from the DEA data*

Description

Generates Visualizations from the DEA data

Usage

```
plotDEResults(
  DE_results_list,
  dataset_name = NULL,
  heatmap = TRUE,
  dim_reduct = NULL,
  numPC = 2,
  target = NULL,
  color_factor = NULL,
  shape_factor = NULL,
  label_factor = NULL,
  highlight_median = FALSE,
  scale_pca = TRUE,
  color_palette = NULL,
  fc_threshold = 1,
  padj_threshold = 0.05,
  label_significant = TRUE,
  show_legend = "none"
)
```

Arguments

DE_results_list	A DESeq2 object with the normalized and vst counts. Can be obtained by running computeDEResults
dataset_name	String to specify the assay in DE_results_list to extract.
heatmap	Logical; if TRUE, generates a heatmap for exploratory analysis. Defaults to TRUE.
dim_reduct	Either "PCA", "UMAP" or "tSNE" to specify the dimensionality reduction approach to be executed. Defaults to "PCA".
numPC	Numeric; number of principal components to include in the PCA analysis. Required if dim_reduct is "PCA". Defaults to 2.
target	Optional; a factor based on metadata columns to define the comparisons to perform in a targeted analysis. Defaults to NULL.
color_factor	Optional; name of the categorical variable in metadata to group by color the data points. Used if dim_reduct specified.
shape_factor	Optional; name of the categorical variable to label the data points. Used if dim_reduct specified.

label_factor Optional; name of the categorical variable to label the data points. Used if dim_reduct specified.

highlight_median Logical; if TRUE the data points of each cluster will be summarized into the median. Defaults to FALSE.

scale_pca Logical; if TRUE, scales the data if dim_reduct is "PCA". Defaults to TRUE.

color_palette Optional; a vector of color codes to customize the plot appearance.

fc_threshold Numeric; fold change threshold used for highlighting significant features in the volcano plot (if targets is specified). Defaults to 1.

padj_threshold Numeric; p-value threshold used for highlighting significant features in the volcano plot. Defaults to 0.05.

label_significant Logical; if TRUE displays the axis of the plots based on fc_threshold and padj_threshold. Defaults to TRUE.

show_legend Either "none" (default), "top", "bottom", "right" or "left" to specify the position of the legend in the plot.

Value

Visualization of the DEA results.

Examples

```
data("default_tTEscanR_tRNA_data", package = "tTEscanR")
data("default_tTEscanR_metadata", package = "tTEscanR")

DE_analysis <- computeDEResults(
    list_data = list(tRNA = default_tTEscanR_tRNA_data),
    metadata = default_tTEscanR_metadata, batch = "tissue",
    compute_pairwise = FALSE
)
DE_plots <- plotDEResults(
    DE_results_list = DE_analysis, dataset_name = "tRNA",
    dim_reduct = "PCA", color_factor = "tissue", heatmap = FALSE
)
```

---

plotDistribution *Distribution Plot of Codon/Anticodon Usage or Amino Acid Demand/Supply*

Description

This function generates a visualization of codon-anticodon usage or amino acid demand-supply distributions across conditions. The input data (data) is expected to be in long format and contain a minimum information regarding the features, the conditions they belong to and their usage counts. Based on the selected plot parameter the user can select the most convenient layout to display the data. It is crucial to ensure consistency between the name of the columns in data and the parameters x_axis_col, y_axis_col and condition_col.

Usage

```
plotDistribution(
  data,
  plot = "jitter",
  bar_position = "dodge",
  x_axis_col,
  y_axis_col,
  condition_col,
  color_palette = NULL,
  ncols = 1,
  facet_col = NULL,
  add_stats = FALSE,
  targeted_arg = NULL,
  free_scale = TRUE,
  filename = NULL,
  show_legend = "none",
  add_titles = TRUE
)
```

Arguments

data	A long format table. This format can be obtained using transformFormat.
plot	Either "jitter" (default), "barplot", "boxplot" or "dot" to indicate the type of plot to generate.
bar_position	Either "dodge" (default), "stack" or "fill" to indicate the display of the plot if plot is "barplot".
x_axis_col	Name of the categorical variable to reflect in the plot.
y_axis_col	Name of the numerical variable to reflect in the plot.
condition_col	Name of the categorical variable to group the data points.
color_palette	Optional; a vector of color codes to customize plot appearance.
ncols	Numeric; number of columns for arranging panels. Defaults to 1.
facet_col	Optional; name of the categorical variable to divide the plot into different panels. Required if ncols bigger than 1.
add_stats	Logical; if TRUE, performs a statistical analysis based on the available parameters. Defaults to FALSE.
targeted_arg	Optional; a vector defining key feature clusters to highlight or label.
free_scale	Logical; if TRUE, enables the scales of the plot to have get different scales. Defaults to TRUE.
filename	Optional; name for the plot to be saved.
show_legend	Either "none" (default), "top", "bottom", "right" or "left" to specify the position of the legend.
add_titles	Logical; if TRUE, includes titles in the plot. Defaults to TRUE.

Value

A ggplot object representing the requested distribution plot. If save_format is provided, the plot will also be saved to the specified location.

Examples

```
data("default_tTEscanR_mRNA_data", package = "tTEscanR")
data("default_tTEscanR_metadata", package = "tTEscanR")

# Define the object and compute the codon usage
tTEscanR_obj <- createObject(
    counts = default_tTEscanR_mRNA_data,
    assay = "mRNA",
    meta_data = default_tTEscanR_metadata,
    sample_id = "conditions",
    params = list("CorrectionFactor" = "tissue")
)
tTEscanR_obj <- computeCodonUsage(
    object = tTEscanR_obj, species = "hg38",
    additional_metrics = FALSE, reduce = 1000
)
codon_usage <- SummarizedExperiment::assay(tTEscanR_obj, "CodonUsage")

# Transform the data
long_cu <- transformFormat(
    data = codon_usage, normalize = TRUE,
    rownames_to_column = "codon", names_to = "condition", values_to = "usage"
)
long_cu <- long_cu |>
    tidyr::separate(condition, into = c("tissue", "cell_type"), sep = "-")

# Generate the plot
codon_usage_plot <- plotDistribution(
    data = long_cu, plot = "jitter", x_axis_col = "codon",
    y_axis_col = "usage", condition_col = "tissue", show_legend = "right",
    add_titles = FALSE
)
```

---

plotPermutation *Permutation Plot*

Description

This function generates a plot to compare the baseline codon exonic background against the current codon usage. For a better interpretation, the codons are colored by amino acid.

Usage

```
plotPermutation(
  permut_data,
  sig_data,
  color_palette = NULL,
  filename = NULL,
  show_legend = "none",
  add_titles = TRUE
)
```

Arguments

permut_data	A table with the codons and their frequencies after computing all the permutations. Output from getPermutationDist.
sig_data	A table with the codon exonic background and their significance level before (p-value) and after the correction (p-adjusted value). Output from obtainSignificance
color_palette	Optional; a vector of color codes to customize plot appearance.
filename	Optional; name for the plot to be saved.
show_legend	Either "none" (default), "top", "bottom", "right" or "left" to specify the position of the legend in the plot.
add_titles	Logical; if TRUE, includes titles in the plot. Defaults to TRUE.

Value

Permutation plot.

Examples

```
data("default_tTEscanR_mRNA_data", package = "tTEscanR")
data("default_tTEscanR_metadata", package = "tTEscanR")

selected_genes <- default_tTEscanR_mRNA_data[1:20, ]
permutation_test <- getPermutationDist(
    n_permut = 100, target_data = selected_genes, species = "hg38"
) # Generate table with codon and freq
tTEscanR_obj <- createObject(
    counts = default_tTEscanR_mRNA_data, assay = "mRNA",
    meta_data = default_tTEscanR_metadata,
    sample_id = "conditions",
    params = list("CorrectionFactor" = "tissue")
)
tTEscanR_obj <- computeCodonUsage(
    object = tTEscanR_obj, species = "hg38",
    additional_metrics = FALSE, reduce = 1000
)

codon_usage <- SummarizedExperiment::assay(tTEscanR_obj, "CodonUsage")
codon_background <- rowSums(codon_usage) / sum(rowSums(codon_usage))
codons_to_AA <- featuresToAA(
    data = names(codon_background),
    notation_from = "codon", notation_to = "aa"
)
codon_background <- data.frame(
    group = codons_to_AA, codon = names(codon_background),
    freq = as.numeric(codon_background), row.names = NULL
)
significance <- obtainSignificance(
    dist = permutation_test, value = codon_background
)

plotPermutation(permut_data = permutation_test, sig_data = significance)
```

plotProportion *Proportion Plot of Codon/Anticodon Usage or Amino Acid Demand/Supply*

Description

This function generates a proportion plot to compare codon-anticodon usage or amino acid demand-supply frequencies across conditions. The input data (`data`) is expected to be in long format and contain a minimum information regarding the features, the conditions they belong to and their usage counts. Based on the selected `plot` parameter the user can select the most convenient layout to display the data. It is crucial to ensure consistency between the name of the columns in `data` and the parameters `var_numerical`, `var_categorical` and `var_color`.

Usage

```
plotProportion(
  data,
  plot = "bar",
  var_numerical,
  var_categorical,
  var_color = NULL,
  facet_col = NULL,
  color_palette = NULL,
  num_limits = NULL,
  num_rings = 5,
  filename = NULL,
  zoom = FALSE,
  show_legend = "none",
  add_titles = TRUE,
  order = NULL,
  normalize = TRUE
)
```

Arguments

`data`	A long format table. This format can be obtained using `transformFormat`.
`plot`	Either `"bar"` (default), `"donut"` or `"radar"` to indicate the type of plot to generate.
`var_numerical`	Name of the numerical variable to reflect in the plot.
`var_categorical`	Name of the categorical variable to reflect in the plot.
`var_color`	Optional; name of the categorical variable to group the data points when coloring. Required if `plot` is `"bar"`.
`facet_col`	Optional; name of the categorical variable to divide the plot into different panels.
`color_palette`	Optional; a vector of color codes to customize plot appearance.
`num_limits`	Optional; a vector with the upper and lower ranges of the values in `var_numerical`.
`num_rings`	Optional; a number specifying the amount of rings to display if `plot` is `"radar"`. Defaults to 5.

filename	Optional; name for the plot to be saved.
zoom	Logical; if TRUE, centers the plot display to the values in data.Defaults to TRUE.
show_legend	Either "none" (default), "top", "bottom", "right" or "left" to specify the position of the legend.
add_titles	Logical; if TRUE, includes titles in the plot. Defaults to TRUE.
order	Optional; a vector of the levels to organize the data, based on the var_categorical.
normalize	Logical; if TRUE, normalizes the data in order to display the relative contribution of the var_categorical. Defaults to TRUE.

Value

A ggplot object representing the requested proportion plot. If save_format is provided, the plot will also be saved to the specified location.

Examples

```
data("default_tTEscanR_mRNA_data", package = "tTEscanR")
data("default_tTEscanR_metadata", package = "tTEscanR")

# Define the object and compute the codon usage
tTEscanR_obj <- createObject(
    counts = default_tTEscanR_mRNA_data, assay = "mRNA",
    meta_data = default_tTEscanR_metadata,
    sample_id = "conditions",
    params = list("CorrectionFactor" = "tissue")
)
tTEscanR_obj <- computeCodonUsage(
    object = tTEscanR_obj, species = "hg38",
    additional_metrics = TRUE, reduce = 1000
)

# Compute and extract the mean codon usage
meta <- S4Vectors::metadata(tTEscanR_obj)
additional_metrics <- meta[["CodonUsage_AdditionalMetrics"]]
mean_codon_usage <- additional_metrics$MeanCodonUsage
mean_codon_usage$codon <- mean_codon_usage$feature

# Translate the codons to amino acids
mean_codon_usage <- featuresToAA(
    data = mean_codon_usage, position = "feature",
    notation_from = "codon", notation_to = "aa"
)

# Generate the plot
plotProportion(
    data = mean_codon_usage, plot = "bar",
    var_numerical = "mean_usage_across_conditions",
    var_categorical = "codon", var_color = "feature", show_legend = "none"
)
```

plotTargetComparison *Distribution Plot of Codon/Anticodon Usage or Amino Acid Demand/Supply Compared to a Target*

Description

This function generates a distribution plot to compare distribution usages between a targeted set of conditions against the rest. A common application of this plot is to compare mean usages across conditions. Both input data sources (`target_data` and `overall_data`) are expected to be in long format and contain a minimum information regarding the features, the conditions they belong to and their usage counts. It is crucial to ensure consistency between the name of the columns in `target_data` and `overall_data`, together with the parameters `x_axis_col` and `y_axis_col`.

Usage

```
plotTargetComparison(
  target_data,
  overall_data,
  x_axis_col,
  y_axis_col,
  color_palette = NULL,
  show_difference = TRUE,
  add_titles = TRUE,
  filename = NULL,
  show_legend = "none"
)
```

Arguments

`target_data`	A long format table of a condition of interest. This format can be obtained using `transformFormat`.
`overall_data`	A long format table of the whole set of conditions (with or without the data in `target_data`. This format can be obtained using `transformFormat`.
`x_axis_col`	Name of the categorical variable to reflect in the plot.
`y_axis_col`	Name of the numerical variable to reflect in the plot.
`color_palette`	Optional; a vector of color codes (min. 2 codes and max. 3 codes) to customize plot appearance. Colors for (i) data, (ii) target value, and (iii) difference bar.
`show_difference`	Logical; if `TRUE`, displays the difference between `overall_data` and `target_data`.
`add_titles`	Logical; if `TRUE`, includes titles in the plot. Defaults to `TRUE`.
`filename`	Optional; name for the plot to be saved.
`show_legend`	Either `"none"` (default), `"top"`, `"bottom"`, `"right"` or `"left"` to specify the position of the legend.

Value

A `ggplot` object representing the requested distribution plot. If `save_format` is provided, the plot will also be saved to the specified location.

plotTEscore *Violin Plot Displaying tTE Scores*

Description

This function generates a violin plot to visualize the distribution of tTE scores across different condition. The tTE scores should be obtained using computeTheoreticalTE. The plot allows the user to easily compare the tTE distribution between different conditions, with the option to highlight specific feature clusters based on the provided targets.

Usage

```
plotTEscore(
  data,
  metadata,
  class_col,
  index_col,
  target_col = NULL,
  score_col = "tTE",
  cond_col = "condition",
  pval_col = "p_value",
  facet_col = NULL,
  color_palette = NULL,
  add_stats = TRUE,
  filename = NULL,
  show_legend = "none",
  add_titles = TRUE,
  show_outliers = FALSE
)
```

Arguments

data	A tTE results table obtained from computeTheoreticalTE.
metadata	A table with additional information regarding the conditions in data.
class_col	Name of the categorical variable to reflect in the plot.
index_col	Name of the categorical variable that links the conditions in data with the metadata.
target_col	Optional; name of the categorical variable to perform the statistical comparison (the most specific level). Used if add_stats is TRUE.
score_col	Name of the numerical variable that contains the tTE scores in the data. Defaults to "tTE".
cond_col	Name of the categorical variable that contains the conditions in the data. Defaults to "condition".
pval_col	Name of the numerical variable that contains the significance scores in data. Defaults to "p_value".
facet_col	Optional; name of the categorical variable separate the plot into different panels.
color_palette	Optional; a vector of color codes to customize plot appearance.

add_stats	Logical; if TRUE, performs a statistical analysis based on the available parameters. Defaults to TRUE.
filename	Optional; name for the plot to be saved.
show_legend	Either "none" (default), "top", "bottom", "right" or "left" to specify the position of the legend in the plot.
add_titles	Logical; if TRUE, includes titles in the plot. Defaults to TRUE.
show_outliers	Logical; if TRUE, labels the outlier data points based on index_col.

Value

A ggplot object representing the tTE scores. If save_format is provided, the plot will also be saved to the specified location. If add_stats reports a table with the statistical measures summarized.

Examples

```
data("default_tTEscanR_mRNA_data", package = "tTEscanR")
data("default_tTEscanR_tRNA_data", package = "tTEscanR")
data("default_tTEscanR_metadata", package = "tTEscanR")

# Define the tTEscanR object
tTEscanR_obj <- createObject(
    counts = list(
        mRNA = default_tTEscanR_mRNA_data, tRNA = default_tTEscanR_tRNA_data
    ),
    meta_data = default_tTEscanR_metadata,
    sample_id = "conditions",
    params = list("CorrectionFactor" = "tissue")
)

# Compute the codon and anticodon usage
tTEscanR_obj <- computeCodonUsage(
    object = tTEscanR_obj, species = "hg38",
    additional_metrics = FALSE, reduce = 10000
)
tTEscanR_obj <- computeAnticodonUsage(object = tTEscanR_obj)

# Compute the theoretical translation efficiency (tTE scores)
tTEscanR_obj <- computeTheoreticalTE(
    object = tTEscanR_obj, level = "codon", compute_significance = TRUE
)
tTE_results <- S4Vectors::metadata(tTEscanR_obj)[["tTE_results"]]
tTEresults_codon <- tTE_results[["tTEresults_codon"]]
conditions_metadata <- as.data.frame(
    SummarizedExperiment::colData(tTEscanR_obj)
)

# Visualize the tTE scores
plotTEscore(
    data = tTEresults_codon, metadata = conditions_metadata,
    index_col = "conditions", class_col = "tissue", add_stats = TRUE
)
```

runDEAnalysis *Perform Differential Expression Analysis Using DESeq2*

Description

This function applies differential expression analysis using the DESeq2 framework on a matrix of expression values. It supports both exploratory visualizations (heatmap and PCA) and targeted comparisons using a custom contrast table.

Usage

```
runDEAnalysis(
  list_data,
  metadata,
  batch = NULL,
  reference = NULL,
  reduce = 100,
  dim_reduct = NULL,
  numPC = 2,
  color_factor = batch,
  color_palette = NULL,
  heatmap = TRUE,
  shape_factor = NULL,
  label_factor = NULL,
  target = NULL,
  highlight_median = FALSE,
  fc_threshold = 1,
  show_legend = "none",
  padj_threshold = 0.05,
  label_significant = TRUE,
  compute_pairwise = TRUE
)
```

Arguments

list_data	A list of matrices with features (i.e. genes, codons, anticodons or amino acids) as rows and samples or conditions as columns. The list can be named or unnamed.
metadata	A data.frame with the data associated to the conditions in the matrices of list_data. There has to be one column with the same labels as the column names.
batch	Optional; name of the categorical variable in metadata to correct the data.
reference	Optional; factor from the batch variable to use as reference for the corrections. If not specified, the 1st factor that appears will be used instead.
reduce	Numeric; a scaling factor used to normalize large expression values that exceed R's handling capacity. Defaults to 100.
dim_reduct	Either "PCA", "UMAP" or "tSNE" to specify the dimensionality reduction approach to be executed. Defaults to "PCA".
numPC	Numeric; number of principal components to include in the PCA analysis. Required if dim_reduct is "PCA". Defaults to 2.

color_factor Optional; name of the categorical variable in metadata to group by color the data points. Used if dim_reduct specified.

color_palette Optional; a vector of color codes to customize the plot appearance.

heatmap Logical; if TRUE, generates a heatmap for exploratory analysis. Defaults to TRUE.

shape_factor Optional; name of the categorical variable in metadata to group by shape the data points. Used if dim_reduct specified.

label_factor Optional; name of the categorical variable to label the data points. Used if dim_reduct specified.

target Optional; a factor based on metadata columns to define the comparisons to perform in a targeted analysis. Defaults to NULL.

highlight_median
Logical; if TRUE the data points of each cluster will be summarized into the median. Defaults to FALSE.

fc_threshold Numeric; fold change threshold used for highlighting significant features in the volcano plot. Defaults to 1.

show_legend Either "none" (default), "top", "bottom", "right" or "left" to specify the position of the legend in the plot.

padj_threshold Numeric; p-value threshold used for highlighting significant features in the volcano plot. Defaults to 0.05.

label_significant
Logical; if TRUE displays the axis of the plots based on fc_threshold and padj_threshold. Defaults to TRUE.

compute_pairwise
Logical; if TRUE, computes all the pairwise comparisons based on the conditions included in the input data.

Value

A list of outputs per each matrix in list_data, based on the enabled parameters: (i) exploratory plots (heatmap and/or PCA), (ii) targeted plot (volcano), and (iii) size-corrected data.

Examples

```
data("default_tTEscanR_tRNA_data", package = "tTEscanR")
data("default_tTEscanR_metadata", package = "tTEscanR")
DE_analysis <- runDEAnalysis(
    list_data = list(tRNA = default_tTEscanR_tRNA_data),
    metadata = default_tTEscanR_metadata, batch = "tissue",
    color_factor = "tissue", compute_pairwise = FALSE
)
```

---

runPipeline *Runs the Theoretical Translation Efficiency (tTE) Pipeline*

Description

This function wraps up all the independent functions of theoretical translation efficiency pipeline. Requires an mRNA and tRNA count matrices to compute the codon and anticodon usage and further derive the amino acid demand and supply. With matching condition in the former matrices the theoretical translation efficiency would be computed.

Usage

```
runPipeline(
  mRNA_data,
  tRNA_data,
  meta_data,
  batch,
  sample_id = NULL,
  corr_method = "spearman",
  additional_metrics = TRUE,
  runDESeq = TRUE,
  compute_significance = TRUE,
  codon_freq = NULL,
  species = NULL,
  genetic_code = "Standard",
  reduce = 100,
  ...
)
```

Arguments

mRNA_data	A count matrix of mRNA genes (rows) per conditions (columns).
tRNA_data	A count matrix of tRNA genes (rows) per conditions (columns).
meta_data	A data.frame with the meta-information related with the conditions in mRNA_data or tRNA_data. Just the intersecting conditions will be considered.
batch	A factor based on meta_data columns to define the variable to correct for when size correcting the count matrices.
sample_id	Optional; a character string naming the column in meta.data to use as sample row names.
corr_method	A correlation method accepted by cor. Defaults to "spearman".
additional_metrics	Logical; if TRUE, computes: (i) codon exonic background, (ii) mean codon usage, and (iii) correlation between the previous metrics. Defaults to TRUE.
runDESeq	Logical; if TRUE, performs differential expression analysis to each assay in the tTEscanR_Object. Defaults to TRUE
compute_significance	Logical; if TRUE, computes the statistical significance (p-value) of the tTE scores. Defaults to TRUE.
codon_freq	Optional; a user-provided codon frequency-per-gene table. If necessary, it can be computed using getCodonFreq.
species	Either "hg38" (human) or "mm39" (mouse) to specify which default codon frequency-per-gene table to use. Required if codon_freq not provided or if the gene annotation is inconsistent between both inputs.
genetic_code	A character string to specify the genetic code to be used. Defaults to "Standard".
reduce	Numeric; a scaling factor used to normalize large expression values that exceed R's handling capacity. Defaults to 100.
...	Additional arguments passed directly to runDEAnalysis (e.g., dim_reduct, compute_pairwise, color_factor).

Value

A MultiAssayExperiment with the assays and metadata computed through the tTE pipeline.

Examples

```
data("default_tTEscanR_mRNA_data", package = "tTEscanR")
data("default_tTEscanR_tRNA_data", package = "tTEscanR")
data("default_tTEscanR_metadata", package = "tTEscanR")

tTEscanR_obj <- runPipeline(
    mRNA_data = default_tTEscanR_mRNA_data,
    tRNA_data = default_tTEscanR_tRNA_data,
    meta_data = default_tTEscanR_metadata,
    species = "hg38", batch = "tissue", additional_metrics = FALSE,
    sample_id = "conditions", compute_significance = FALSE, runDESeq = FALSE
)
```

---

showPoolContribution *Examine the Codon Pool Contribution*

Description

This function analyzes the contribution of the most highly expressed genes to the overall codon pool across conditions. It is particularly useful for evaluating codon bias in highly expressed genes and how it varies across conditions. If needed, gene annotations can be translated for consistency, and internal species-specific for human ("hg38") and mouse ("mm39") are supported.

Usage

```
showPoolContribution(
  object,
  codon_freq = NULL,
  species = NULL,
  N = 10,
  corr_method = "spearman",
  overwrite = FALSE
)
```

Arguments

object	A MultiAssayExperiment containing a "mRNA" and "CodonUsage" assay.
codon_freq	Optional; a user-provided codon frequency per gene table. If necessary, it can be computed using getCodonFreq.
species	A character string specifying the species reference genome version (used if codon_freq is not provided or translate is TRUE). Supported values include "hg38" (human) and "mm39" (mouse).
N	Numeric; number of top genes to consider in the codon pool contribution. Defaults to 10.
corr_method	A correlation method accepted by cor. Defaults to "spearman".
overwrite	Logical; if TRUE, overwrites any existing anticodon usage assay and metadata in the object. Defaults to FALSE.

Value

An updated MultiAssayExperiment containing new layers of information in the params slot, representing the codon pool contribution, "CodonPoolContribution_Results".

Examples

```
data(default_tTEscanR_mRNA_data, default_tTEscanR_metadata)
tTEscanR_obj <- createObject(
    counts = list(mRNA = default_tTEscanR_mRNA_data),
    meta_data = default_tTEscanR_metadata, sample_id = "conditions",
    params = list("CorrectionFactor" = "tissue")
)
tTEscanR_obj <- computeCodonUsage(
    object = tTEscanR_obj, species = "hg38", additional_metrics = FALSE
)
tTEscanR_obj <- showPoolContribution(
    object = tTEscanR_obj, species = "hg38"
)
```

---

transformFormat *Transform the format of a table*

Description

This function converts a matrix or data.frame into a tidy, long-format tibble. Optionally normalizes the values.

Usage

```
transformFormat(data, normalize, rownames_to_column, names_to, values_to)
```

Arguments

data	A table to be converted. Supported formats: matrix or data.frame.
normalize	Logical; if TRUE, values are converted to relative frequencies. Defaults to FALSE.
rownames_to_column	A character string specifying the name of the new column that will hold the former row names in data.
names_to	A character string specifying the name of the new column that will hold the former column names in data.
values_to	A character string specifying the name of the new column that will hold the corresponding values from the pivoted columns in data.

Value

A tibble of the input data

Examples

```
data("default_tTEscanR_tRNA_data", package = "tTEscanR")

tRNA_long_format <- transformFormat(
    data = default_tTEscanR_tRNA_data,
    normalize = FALSE,
    rownames_to_column = "tRNA_genes",
    names_to = "condition",
    values_to = "abundance"
)
tTEobj <- createObject(
    counts = default_tTEscanR_tRNA_data,
    assay = "tRNA"
)
tTEobj <- computeAnticodonUsage(object = tTEobj)
anticodon_long_format <- transformFormat(
    data = SummarizedExperiment::assay(tTEobj, "AnticodonUsage"),
    normalize = TRUE,
    rownames_to_column = "anticodons",
    names_to = "condition", values_to = "usage"
)
```

---

tRNAFilterCuts *Filter Out Conditions With Low tRNA Cuts*

Description

This function filters a tRNA expression matrix by removing conditions (columns) that fall below a specific total read count (`cutoff`). It is useful for eliminating low-quality or poorly sequenced conditions that may bias downstream analyses.

Usage

```
tRNAFilterCuts(data, cutoff = 5000)
```

Arguments

`data`	A `matrix` or `data.frame` of tRNA gene expression data, with tRNA genes as rows and conditions as columns.
`cutoff`	Numeric; minimum total number of tRNA cuts required to retain a condition in `data`. Defaults to 5000.

Value

A filtered `matrix` or `data.frame` with tRNAs below the cutoff removed.

Examples

```
data("default_tTEscanR_tRNA_data", package = "tTEscanR")

tRNA_data_filtered <- tRNAFilterCuts(
    data = default_tTEscanR_tRNA_data,
    cutoff = 5000
)
```

`tRNAGetMatrix` *Generate a tRNA expression matrix*

Description

Generate a tRNA expression matrix

Usage

```
tRNAGetMatrix(
  data,
  assay = "peaks",
  confidence_set = NULL,
  tRNA_name_map = NULL,
  species = NULL,
  flanking_region = 100,
  name_sep = c("-", "-"),
  filename = NULL
)
```

Arguments

`data`	`SummarizedExperiment`, `ChromatinAssay`, or `SeuratObject`.
`assay`	Optional; a character string specifying the name of the assay to retrieve from `data` if it is a `SeuratObject`. Defaults to `"peaks"`.
`confidence_set`	Either a file path to the tRNA annotations (confidence set file from gtRNAdb), or a GRanges object. Contains the set of high confidence tRNA genes
`tRNA_name_map`	Optional; a `data.frame` with the tRNA gene names linked to `confidence_set`. Contains two columns: tRNAscan-SE and GtRNAdb gene ids.
`species`	Optional; either `"hg38"` (human) or `"mm39"` (mouse) to load the default `confidence_set`
`flanking_region`	Integer; number of nucleotides that form the flanking region of each tRNA. Defaults to 100.
`name_sep`	A string delimiter to format the tRNA gene names in the output matrix. Defaults to `c("-", "-")`.
`filename`	Optional; name for the tRNA matrix to be saved.

Value

Sparse matrix of tRNA counts (tRNAs x cells)

tRNASetCutoff *Selection of the Optimal tRNA Cut Cutoff*

Description

Selection of the Optimal tRNA Cut Cutoff

Usage

```
tRNASetCutoff(
  data,
  num_iter = 1000,
  cutoffs_limits = c(50, 10000),
  generate_plot = TRUE,
  slope_threshold = 0.001,
  rho_threshold = 0.95,
  compute_aa = FALSE
)
```

Arguments

data	tRNA gene expression count matrix with tRNA genes as rows and conditions as columns.
num_iter	Numeric; value to select the number of iterations to perform in order to determine the optimal cutoff. Defaults to 1000.
cutoffs_limits	Minimum and maximum values to test to search for the optimal tRNA cuts threshold. Defaults to c(50, 10000).
generate_plot	Logic; if TRUE, generates a correlation plot. Defaults to TRUE.
slope_threshold	Numeric; value to consider for the determination of the correlation stability. Defaults to 0.001.
rho_threshold	Numeric; value to consider for the determination of the correlation strength. Defaults to 0.95.
compute_aa	Logic; if TRUE, computes the amino acid supply, otherwise only considers the anticodon usage. Defaults to FALSE.

Value

Table with the optimal cutoff at the anticodon isoacceptor and amino acid isotype.

Examples

```
data("default_tTEscanR_tRNA_data", package = "tTEscanR")

optimal_tRNA_cutoffs <- tRNASetCutoff(
    data = default_tTEscanR_tRNA_data,
    generate_plot = FALSE, num_iter = 10,
    cutoffs_limits = c(3500, 4000)
)
```

tRNASetGenes *Annotate the tRNA genes from tRNA tags*

Description

Annotate the tRNA genes from tRNA tags

Usage

```
tRNASetGenes(data, tRNA_bed, flanking_region = 100, name_sep = c("-", "-"))
```

Arguments

data	A matrix with tRNA tags as rows.
tRNA_bed	Path to the directory that contains the .bed file
flanking_region	Numeric; number of bases to include expand the region interrogated. Defaults to 100.
name_sep	A string delimiter to format the tRNA gene names in the output matrix. Defaults to c("-", "-").

Value

A data with the translated tRNA gene names.

updateObject *Update Assay in a tTEscanR object*

Description

Updates an existing MultiAssayExperiment object by adding or replacing one or more count matrices, sample metadata, or global parameters.

Usage

```
updateObject(
  object,
  counts = NULL,
  assay = NULL,
  meta_data = NULL,
  sample_id = NULL,
  params = NULL,
  overwrite = FALSE
)
```

Arguments

object	An existing tTEscanR_Object.
counts	A count matrix or data.frame to append or update.
assay	Character string specifying the target assay name.
meta_data	Optional; a data.frame of sample metadata
sample_id	Optional; a character string naming the column in meta.data to use as sample row names.
params	Optional; a list of global run parameters.
overwrite	Logical; if TRUE, overwrites existing assays or metadata if the label coincides. Defaults to FALSE.

Value

An updated MultiAssayExperiment object.

Examples

```
data("default_tTEscanR_mRNA_data", package = "tTEscanR")
data("default_tTEscanR_tRNA_data", package = "tTEscanR")

tTEscanR_obj <- createObject(
    counts = default_tTEscanR_mRNA_data,
    assay = "mRNA",
    sample_id = "conditions"
)
tTEscanR_obj <- updateObject(
    object = tTEscanR_obj,
    counts = default_tTEscanR_tRNA_data,
    assay = "tRNA"
)
```

Index

∗ **datasets**
default_tTEscanR_metadata, 14
default_tTEscanR_mRNA_data, 14
default_tTEscanR_tRNA_data, 15
∗ **internal**
tTEscanR-package, 3

computeAAUsage, 4
computeAnticodonUsage, 5
computeCodonUsage, 5, *21*
computeCorrelationBackground, 7
computeDEResults, 8, *24*
computeExonicBackground, *7*, 9, *21*
computeMeanUsage, *7*, 10
computeTheoreticalTE, 11, *32*
controlVerbose, 12
cor, *6, 7*, *11*, *36, 37*
createObject, *10*, 13

default_tTEscanR_metadata, 14
default_tTEscanR_mRNA_data, 14
default_tTEscanR_tRNA_data, 15

extractCodons, 16

featuresToAA, 16

getCodonFreq, *6*, 17, *19*, *36, 37*
getPermutationDist, 19, *21*, *28*
groupConditions, 20

mergeMatrices, 20
MultiAssayExperiment, *4–6*, *11*, *13*, *37, 38*, *42, 43*

obtainSignificance, 21, *28*

plotCorrelation, 22
plotDEResults, 24
plotDistribution, 25
plotPermutation, 27
plotProportion, 29
plotTargetComparison, 31
plotTEscore, 32

runDEAnalysis, 34, *36*
runPipeline, 35

showPoolContribution, 37

transformFormat, *23*, *26*, *29*, *31*, 38
tRNAFilterCuts, 39
tRNAGetMatrix, 40
tRNASetCutoff, 41
tRNASetGenes, 42
tTEscanR *(*tTEscanR-package*)*, 3
tTEscanR-package, 3

updateObject, 42