Package 'mobileRNA'

November 12, 2026

Type Package

Title mobileRNA: Investigate the RNA mobilome & population-scale changes

Version 1.8.0

Description Genomic analysis can be utilised to identify differences between RNA populations in two conditions, both in production and abundance. This includes the identification of RNAs produced by multiple genomes within a biological system. For example, RNA produced by pathogens within a host or mobile RNAs in plant graft systems. The mobileRNA package provides methods to pre-process, analyse and visualise the sRNA and mRNA populations based on the premise of mapping reads to all genotypes at the same time.

License MIT + file LICENSE

Depends R (>= 4.3.0)

VignetteBuilder knitr

Encoding UTF-8

LazyData FALSE

Roxygen list(markdown = TRUE)

RoxygenNote 7.3.3

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

Imports dplyr, tidyr, ggplot2, BiocGenerics, DESeq2, edgeR, ggrepel, grDevices, pheatmap, utils, tidyselect, progress, RColorBrewer, GenomicRanges, rtracklayer, data.table, SimDesign, scales, IRanges, stats, methods, Biostrings, reticulate, S4Vectors, GenomeInfoDb, SummarizedExperiment, rlang, bioseq, grid

biocViews Visualization, RNASeq, Sequencing, SmallRNA, GenomeAssembly, Clustering, ExperimentalDesign, QualityControl, WorkflowStep, Alignment, Preprocessing

BiocType Software

BugReports https://github.com/KJeynesCupper/mobileRNA/issues

SystemRequirements GNU make, ShortStack (>= 4.0), HTSeq, HISAT2, SAMtools, Conda

Config/testthat/edition 3

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

git_branch RELEASE_3_23

git_last_commit 8240325

git_last_commit_date 2026-04-28

Repository Bioconductor 3.23

Date/Publication 2026-11-11

Author Katie Jeynes-Cupper [aut, cre] (ORCID:
<https://orcid.org/0009-0000-1350-1371>),
Marco Catoni [aut] (ORCID: <https://orcid.org/0000-0002-3258-2522>)

Maintainer Katie Jeynes-Cupper <kejc@illinois.edu>

Contents

mapRNA . . . 3
mobileRNA . . . 7
mRNA_data . . . 8
plotHeatmap . . . 9
plotRNAfeatures . . . 10
plotSampleDistance . . . 12
plotSamplePCA . . . 13
RNAattributes . . . 14
RNAdf2se . . . 16
RNAdicercall . . . 17
RNAdifferentialAnalysis . . . 18
RNAdistribution . . . 20
RNAfeatures . . . 22
RNAimport . . . 23
RNAmergeAnnotations . . . 26
RNAmergeGenomes . . . 28
RNAmobile . . . 29
RNApopulation . . . 31
RNAreorder . . . 33
RNAsequences . . . 34
RNAsubset . . . 36
RNAsummary . . . 37
sRNA_data . . . 38

Index **40**

mapRNA *mobileRNA pre-processing method for sRNAseq & mRNAseq (alignment, raw count or cluster analysis)*

Description

The mobileRNA workflow includes specific pre-processing guidelines. For sRNAseq, this undertakes alignment with Bowtie and sRNA cluster analysis with ShortStack. For mRNAseq, this undertakes alignment with HISAT2 and HTSeq. All OS software should be installed within a Conda environment.

Usage

```
mapRNA(
  input = c("mRNA", "sRNA"),
  sampleData = NULL,
  tidy = TRUE,
  input_files_dir,
  output_dir,
  genomefile,
  annotationfile = NULL,
  condaenv,
  threads = 6,
  mmap = "n",
  dicermin = 20,
  dicermax = 24,
  mincov = 0.5,
  pad = 200,
  order = "pos",
  stranded = "no",
  a = 10,
  mode = "union",
  nonunique = "none",
  type = "mRNA",
  idattr = "Name",
  python = "python"
)
```

Arguments

input	string; define type of Next-Generation Sequencing data set. "sRNA" for sRNAseq data and "mRNA" for mRNAseq data.
sampleData	dataframe; stores mRNA sample data where rows represent each sample in the analysis. Column one stores the sample names, while column two stores the the name(s) of the fastq file(s) for mate 1 (e.g. flyA_1.fq, flyB_1.fq) and column three stores the the name(s) of the fastq file(s) for mate 2 (e.g. flyA_2.fq,flyB_2.fq).

	If data is single ended, column three will not hold any values. Only for mRNA analysis.
tidy	logical; removes unnecessary extra output files when set to TRUE.
input_files_dir	path; directory containing only the FASTQ samples for analysis. Note that all samples in this directory will be used by this function.
output_dir	path; directory to store output.
genomefile	path; path to a FASTA genome reference file.
annotationfile	path; path to a GFF file. For mRNA analysis only.
condaenv	character; name or directory of the Conda environment to use where OS dependencies are stored.
threads	numeric; set the number of threads to use where more threads means a faster completion time. Default is 6.
mmap	character; define how to handle multi-mapped reads. Choose from "u", "f" , "r" or "n". For core sRNA analysis, use either "u", "f" or "r" options. Where "u" means only uniquely-aligned reads are used as weights for placement of multi-mapped reads. Where "f" means fractional weighting scheme for placement of multi-mapped reads and "r" mean multi-mapped read placement is random. For core mRNA analysis, to include multimapped reads, use any parameter, other than "n". While for mobile sRNA or mRNA, it is important to use "n", to not consider multi-mapped reads, only unique reads as we cannot distinguish which genome the reads mapped to multiple locations in.
dicermin	integer; the minimum size in nucleotides of a valid small RNA. This option sets the bounds to discriminate dicer-derived small RNA loci from other loci. Default is 20. For sRNA analysis only.
dicermax	integer; the maximum size in nucleotides of a valid small RNA. This option sets the bounds to discriminate dicer-derived small RNA loci from other loci. Default is 24. For sRNA analysis only.
mincov	numeric; minimum alignment depth, in units of reads per million, required to nucleate a small RNA cluster during de novo cluster search. Must be a number > 0. Default is 2. For sRNA analysis only.
pad	integer; initial peaks are merged if they are this distance or less from each other. Must >= 1, default is 75. For sRNA analysis only.
order	character; either "name" or "pos" to indicate how the input data has been sorted. For paired-end data only, this sorts the data either by read name or by alignment position. Default is "pos", to sort by position. For mRNA analysis only.
stranded	whether the data is from a strand-specific assay, either "yes"/"no"/"reverse". Default is "no". For mRNA analysis only.
a	numeric; skip all reads with alignment quality lower than the given minimum value (default: 10). For mRNA analysis only.
mode	character; states mode to handle reads overlapping more than one feature. Either "union", "intersection-strict" and "intersection-nonempty". Default is "union". For mRNA analysis only.

nonunique	character; states the mode to handle reads that align to or are assigned to more than one feature in the overlap. Either "none" and "all". Default is "none". For mobile mRNA, ensure the default is utilized to exclude multimapped reads. For mRNA analysis only.
type	character; feature type (3rd column in GFF file) to be used, all features of other type are ignored. Default is "mRNA". For mRNA analysis only.
idattr	character; GFF attribute to be used as feature ID. Several GFF lines with the same feature ID will be considered as parts of the same feature. The feature ID is used to identity the counts in the output table. Default is "Name". For mRNA analysis only.
python	character; phython version. default is python but could be python3 etc. This is for defining HTseq.

Details

Please ensure all OS software is installed within a Conda environment. See appendix of vignette for manual pipeline. Alignment statistics are reported for each analysis within log plain text files (log.txt).

In order to align reads, the function will check whether a genome reference index has already been generated and, if not, will generate one. The method varies between sRNA and mRNA analysis depending on the alignment tool. This is generated in the same location as the reference file.

NOTE: This function utilises the `reticulate` R package to connect to the conda environment. Hence, restart R if you wish to change the employed Conda environment during a session.

For sRNA analysis The function invokes a number of OS commands, and is dependent on the installation of `ShortStack` (>= 4.0) with Conda. Please note that `ShortStack` is only compatible with Linux and Mac operating systems.

The pipeline undertakes de novo detection of sRNA-producing loci and alignment, where the output of each are stored in their respective folders in the users desired location. The de novo detection of sRNA-producing loci analyses each sample to identify de novo sRNA-producing loci (ie. sRNA clusters), and joins these results into a single file called "locifile.txt". The alignment step aligns and clusters each sample to the genome reference along with the file containing the de novo sRNA clusters. The final reports are imported into R using `RNAimport()`.

For mRNA analysis

The function invokes a number of OS commands, and is dependent on the installation of `HISAT2`,`HTSeq` and `SAMtools` with `Conda`. The pipeline can undertake single- or pair-end analysis, and to do so requires a data frame stating the sample information where each row represents a sample. The reads are mapped using `HISAT` and then the raw counts are estimated by `HTSeq`. The output alignment file (BAM) and raw counts file for each sample are stored within the samples own folder within the desired directory.

Value

** For sRNA analysis** The OS commands generate output into the users desired location, generating two folders:

- 1_de_novo_detection: Stores output from the detection of de novo sRNA-producing loci

- 2_sRNA_results: Stores results

The first folder stores the alignment (BAM) and the de novo sRNA-producing loci for each sample (.txt) within the samples respective folder. The analyses joins the de novo sRNA clusters across the experimental design which is stored in "locifile.txt". The second folder stores the final clustering results for each sample, and as before the results of each sample are stored within it's respective folder. These results (.txt) are imported into R for downstream analysis by utilizing the `RNAimport()` function.

The function generates a number of extra files for each sample and are not required for the downstream analysis. These are generated by `ShortStack`, see documentation for more information (https://github.com/MikeAxtell/ShortStack). As default these files are deleted. This is determined by the `tidy` argument.

** For mRNA analysis** For mRNA analysis, generate a new folder which stores the results in the users desired output location, known as "1_mRNA_preprocessing". Within this folder, there will contain one folder per sample storing it's sorted alignment file (BAM) and raw counts file ("Result.txt"). Note, that the function excludes multi-mapped mRNAs.

References

ShortStack https://github.com/MikeAxtell/ShortStack, HISAT2 https://anaconda.org/bioconda/hisat2, HTSeq https://htseq.readthedocs.io/en/master/install.html, SAMtools https://anaconda.org/bioconda/samtools

Examples

```
## Not run:

## EXAMPLE 1 - sRNAseq
samples <- file.path(system.file("extdata/sRNAseq",package="mobileRNA"))
GenomeRef <- system.file("extdata","reduced_chr12_Eggplant.fa.gz", package="mobileRNA")
output_location <- tempdir()

mapRNA(input = "sRNA",
input_files_dir = samples,
output_dir = output_location,
genomefile = GenomeRef,
condaenv = "ShortStack4",
mmap = "n")


## EXAMPLE 2 - mRNAseq
GenomeRef <- system.file("extdata","reduced_chr12_Eggplant.fa.gz", package="mobileRNA")
GenomeGFF <- system.file("extdata","reduced_chr12_Eggplant.gff.gz", package="mobileRNA")
# create sample data including name, and file mates:
sampleData <- data.frame(sample = c("selfgraft_1", "selfgraft_2",
                                    "heterograft_1", "heterograft_2"),
                   mate1 = c("selfgraft_mRNAdemo_1.fq.gz", "selfgraft_mRNAdemo_2.fq.gz",
                        "heterograft_mRNAdemo_1.fq.gz", "heterograft_mRNAdemo_2.fq.gz"))
# location of samples:
samples <- system.file("extdata/mRNAseq", package="mobileRNA")
```

```
# location to store output
output_location <- tempdir()

# run alignment
mapRNA(input = "mRNA",
      input_files_dir = samples,
      output_dir = output_location,
      genomefile = output_assembly_file,
      annotationfile = output_annotation_file,
      sampleData = sampleData,
      condaenv = "/Users/user-name/miniconda3")

## End(Not run)
```

---

mobileRNA *mobileRNA: Explore RNA mobilome & population-scale changes*

Description

Uses small RNA or messenger RNA sequencing data in two conditions and identifies changes in the RNA population. `mobileRNA` was primarily designed for the identification of a putative RNA mobilome in a chimeric system. For example, in plant graft systems. As input, `mobileRNA` takes sRNAseq or mRNAseq fastq files. Output consists of a data frame with putative differences between two conditions along with a number of plots.

Details

The most important functions in the **mobileRNA** are:

`RNAmergeGenomes` Merge two genome assembly files (FASTA).

`RNAmergeAnnotations` Merge two genome annotation files (GFF).

`mapRNA` Pre-processing of sRNAseq and mRNAseq (alignment, raw count, cluster analysis).

`RNAimport` Reads the pre-processing report files into a dataframe for all conditions

`RNAdicercall` Calculates the consensus sRNA dicercall class.

`RNAsubset` Subsets the data set based on the sRNA class.

`RNAdifferentialAnalysis` Undertakes differential analysis with either the edgeR or DESeq2 method.

`RNAmobile` Identify putative RNA molecules produced by the non-tissue sample genome

`RNApopulation` Identify gained/lost RNA populations between treatment and control conditions.

`RNAsummary` Summarise the differential abundance of RNAs.

`RNAreorder` Reorder the data frame for differential analysis, ensuring control verse treatment comparison.

`RNAsequences` Extract RNA sequence from sRNA clusters.

`RNAattributes` Overlap the genomic features related to the sRNA clusters.

RNAdistribution Plot the distribution of sRNA classes based on nucleotide length.

plotHeatmap Heatmap of log-transformed normalization data.

plotSampleDistance Plots a sample distance heatmap for quality control.

plotSamplePCA Plots a PCA plot, customize ratio, colours and shapes.

RNAfeatures Summarise the distribution of sRNA clusters across genomic features.

RNAdf2se Convert a mobileRNA dataframe to a SummarizedExperiment object.

Author(s)

Katie Jeynes-Cupper <kej031@student.bham.ac.uk>, Marco Catoni <m.catoni@bham.ac.uk>
Maintainer: Katie Jeynes-Cupper <kej031@student.bham.ac.uk>

See Also

See vignette("mobileRNA", package = "mobieRNA") for an overview of the package.

---

mRNA_data *mRNA_data: simulated messenger RNA data for biological replicates*

Description

Simulated mRNAseq dataset

Usage

```
data(mRNA_data)
```

Details

Simulates data is taken from eggplant and tomato mRNAseq samples and created to simulate to movement of mRNA molecules from an Tomato rootstock to an Eggplant Scion. Two Eggplant replicates were spiked with the same 150 tomato mRNA clusters, and named "heterograft_" 1 to 2. The analysis compares these heterografts to two Eggplant self-grafts which are the original un-spiked Eggplant replicates, called "selfgraft_" 1 to 2.

This data was imported and organised by the RNAimport() function.

Value

Dataframe in global environment

Examples

```
data("mRNA_data")
```

plotHeatmap *Heatmap of log-transformed normalization data*

Description

Undertakes normalisation of RPM/FPKM using a pseudocount and transforms the data using log-scale, to enable visualization of the differences and patterns in expression across samples using a heatmap.

Usage

```
plotHeatmap(
  data,
  value = "RPM",
  pseudocount = 1e-06,
 colours = (grDevices::colorRampPalette(RColorBrewer::brewer.pal(9, "GnBu")))(100),
  cluster = TRUE,
  scale = "none",
  clustering_method = "complete",
  row.names = FALSE,
  border.color = NA,
  column.angle = 45,
  title = NA
)
```

Arguments

data	data.frame; originally generated by RNAimport()
value	character; state the values to plot, either FPKM or RPM.
pseudocount	numeric; pseudo count, default is 1e-6
colours	character; colors. Default is grDevices::colorRampPalette(RColorBrewer::brewer.pal(9, "GnBu"))(100)
cluster	logical; include hierarchical clustering when default cluster= TRUE
scale	character; indicating whether the values should be centered & scaled in either the row direction or the column direction, or none. Respective options are "row", "column" & "none". Default is scale="none".
clustering_method	character; clustering method used. Accepts the same values as hclust. Default clustering_method= "complete"
row.names	logical; indicated whether to include rownames from column 1 of data frame. Default row.names=FALSE
border.color	character; the border colour. Default is no border (NA).
column.angle	numeric; angle of the column labels, choose from 0, 45, 90, 270 or 315.
title	character; states plot title, placed at the top center

Details

Undertakes FPKM/RPM normalisation using a pseudocount and then transforms the normalised-RPM data using log-scale.

This function expects to receive a data frame containing FPKM/RPM data. This function employs the use of a pseudo count during normalisation as the function is expected to be used when identifying mobile sRNAs in a chimeric system. In such system, it is expected that control replicates will contain zero values for the candidate mobile sRNA clusters.

Value

Produces a list objects storing the heatmap plot and the data.

Examples

```
data("sRNA_data")

# vector of control names
controls <- c("selfgraft_1", "selfgraft_2" , "selfgraft_3")

# Locate potentially mobile sRNA clusters associated to tomato, no
# statistical analysis
sRNA_data_mobile <- RNAmobile(input = "sRNA", data =  sRNA_data,
controls = controls, genome.ID = "B_", task = "keep", statistical = FALSE)

# plot heatmap of potential mobile sRNAs
 p1 <-  plotHeatmap(sRNA_data_mobile)
```

---

plotRNAfeatures *Plots the distribution of genomic features in the genome and those that overlap with sRNA clusters*

Description

Using the RNAfeatures() function, this function plots the percentage distribution of genomic features in the provided genome annotation and percentage the distibution of genomic features which overlap sRNA clusters. This is illustrated as a stacked bar plot.

Usage

```
plotRNAfeatures(
  data,
  annotation,
  repeats = NULL,
  promoterRegions = 1000,
```

```
  repeat.type = NULL,
  colors = c("#EEB05AFF", "#E35E28FF", "#4DA284FF", "#4692B0FF", "#134B73FF",
    "#B5361CFF"),
  legend.position = "bottom",
  rmFeatures = NULL
)
```

Arguments

data	data.frame; generated by RNAimport()
annotation	path; URL or connection to a GFFFile object. A genome reference annotation file (.gff/.gff1/.gff2/.gff3). Can be in compressed format (gzip).
repeats	path; URL or connection to a GFFFile object. A genome reference annotation file, which only contains information on repeat sequences in the genome (.gff/.gff1/.gff2/.gff3). By default, this is not required, however if there is a specific repeats annotation file for the genome it is suggested to supply it. Can be in compressed format (gzip).
promoterRegions	numeric; defines the upstream promoter region of genes. Default is 1000, which refers to promoters set at 1Kb upstream of genes
repeat.type	character; features type in annotation file to represent repeats when repeats not supplied.
colors	vector; colours for each feature.
legend.position	character; position of legend. Either "none", "left", "right", "bottom", "top", "inside".
rmFeatures	vector; type of genomic features to not consider such as "gene" or "exon" etc.

Details

RNAfeatures calculates the number or percentage of sRNA clusters which overlap with genomic features based on their genomic coordinates.

Value

Returns a table containing the number or percentage of overlaps in the supplied sRNA data set with specific regions in the genome annotation such as genes, repeats, introns, exons.

See Also

RNAmergeAnnotations() to merge 2 GFF files into 1.

Examples

```
data("sRNA_data")
features_plot <- plotRNAfeatures(data = sRNA_data,
                      annotation = system.file("extdata",
                      "reduced_chr2_Tomato.gff.gz", package="mobileRNA"))
```

plotSampleDistance *Sample distance matrix*

Description

Draws a simple hierarchical clustered heatmap to observe sample distance.

Usage

```
plotSampleDistance(
  data,
 colours = (grDevices::colorRampPalette(rev(RColorBrewer::brewer.pal(9, "GnBu"))))(255),
  vst = FALSE,
  cellheight = 40,
  cellwidth = 40
)
```

Arguments

data	data.frame; originally generated by RNAimport()
colours	character; the colour palette. Default is grDevices::colorRampPalette(RColorBrewer::brewer.pal("GnBu"))(100)
vst	logical; to undertake variance stabilizing transformation. By default, the function uses a regularized log transformation on the data set, however, this will not suit all experimental designs.
cellheight	numeric; individual cell height in points. Default is 40.
cellwidth	numeric; individual cell width in points. Default is 40.

Details

In special conditions, regularized log transformation will not suit the experimental design. For example, an experimental design without replicates. In this instance, it is preferable to change the default setting and switch to a variance stabilizing transformation method (vst=TRUE).

Value

A blue/green scale heatmap illustrating the sample distance.

Examples

```
data("sRNA_data")
p1 <- plotSampleDistance(sRNA_data)
```

plotSamplePCA *PCA plot of PC1 and PC2*

Description

Draws a principal component analysis (PCA) plot of PC1 and PC2. The function undertakes rlog transformation of the data in an unbiased manner (`blind=TRUE`).

Usage

```
plotSamplePCA(
  data,
  group,
  vst = FALSE,
  labels = TRUE,
  boxed = TRUE,
  legend.title = "Conditions",
  size.ratio = 2,
  colours = NULL,
  point.shape = TRUE,
  ggplot.theme = NULL,
  label.box.padding = 1,
  title = "PCA plot",
  legend.position = "top",
  legend.direction = "horizontal"
)
```

Arguments

`data`	data.frame; originally generated by `RNAimport()`
`group`	character; contains experimental conditions for each replicate. *IMPORTANT*: Ensure this is in the same order as the replicates are found in the data frame supplied to the `data` argument (from left to right).
`vst`	logical; variance stabilizing transformation. By default, the function uses a regularized log transformation on the data set, however, this will not suit all experimental designs.
`labels`	logical; include sample name labels on PCA. Default `labels=TRUE`
`boxed`	logical; add a box around each sample name label. Default `boxed=TRUE`
`legend.title`	character; title for legend key. Default `legend.title = "Conditions"`
`size.ratio`	numeric; set plot ratio, broadens axis dimensions by ratio. Default `size.ratio=2`, double the plot dimension.
`colours`	character; vector of HEX colour codes. Must match the number of conditions.
`point.shape`	logical; set whether the point shapes should be different for each condition.
`ggplot.theme`	character; state the `ggplot2` theme (without () brackets). For example, `ggplot.theme=ggplot2::theme_`

`label.box.padding`
: numeric; Amount of padding around bounding box, as a unit or number. Defaults to 1.

`title`
: character; title for plot.

`legend.position`
: character; the position of legends ("none", "left", "right", "bottom", "top", or two-element numeric vector)

`legend.direction`
: character; layout of items in legends ("horizontal" or "vertical")

Details

This function uses the `DESeq2` package to organise and plot the data. It organises the data into a DESeqDataSet which undergoes log-transformation where the results are used to undertake the PCA analysis. The results are plotted against the principal components 1 and 2.

In special conditions, regularized log transformation will not suit the experimental design. For example, an experimental design without replicates. In this instance, it is preferable to change the default setting and switch to a variance stabilizing transformation method (`` `vst=TRUE` ``).

Value

A PCA plot to show sample distance.

Examples

```
data("sRNA_data")

groups <- c("Heterograft", "Heterograft", "Heterograft",
            "Selfgraft", "Selfgraft", "Selfgraft")


p <-  plotSamplePCA(data = sRNA_data, group = groups)

plot(p)
```

`RNAattributes` *Overlap the genomic features related to the sRNA clusters*

Description

Overlap the genomic features related to the sRNA clusters

Usage

```
RNAattributes(
  data,
  annotation,
  match = c("within", "genes"),
  bufferRegion = 1000
)
```

Arguments

data	data.frame; originally generated by RNAimport() or containing chr, start and end columns.
annotation	path; URL, connection or GFFFile object. A genome reference annotation file (.gff/.gff1/.gff2/.gff3).
match	character; must be either "within" or "genes". Where "within" will return matches where the clusters can be found within any annotation, while "genes" will return matches where the clusters can be found within only genes.
bufferRegion	numeric; a buffer region in base-pairs to extend the start and end coordinates upstream and downstream respectively.

Details

Based on genomic coordinates, assign sRNA clusters with matching annotation information. This function can be used to find the genomic features from which the sRNA clusters originate from. This includes genes or repetitive regions. An additional buffer region at the start/end of the gene is added to improve hits, and align with the assumptions about promoter regions.

It is important that any alteration which were made to the genome reference (FASTA) used for alignment/clustering, such as alterations to the chromosome name, must be carried forth to the genome annotation file. See RNAmergeGenomes() and RNAmergeAnnotations() for more information.

Value

Appends the attribute columns from the GFF file to the supplied data based on overlapping genomic regions.

Examples

```
 # load data
data("sRNA_data")

attributes_df <- RNAattributes(data = sRNA_data,
                    annotation = system.file("extdata",
                    "prefix_reduced_chr2_Tomato.gff", package="mobileRNA"),
                    match = "genes")
```

RNAdf2se *Convert a* mobileRNA *dataframe to a SummarizedExperiment object*

Description

Convert any mobileRNA output dataframe into a SummarizedExperiment object.

Usage

```
RNAdf2se(input = c("sRNA", "mRNA"), data)
```

Arguments

input	character; must be either "sRNA" or "mRNA"
data	data.frame produced by the **mobileRNA** package.

Details

The function relies on the naming structure of columns created by functions in the **mobileRNA** package. It is able to extract the sample names based on these additions, and organise the data appropriately.

Value

A SummarizedExperiment object containing information from working data frame.

#'**For sRNAseq data**

- The rownames contain the locus name and the cluster name.
- The assays represent the additional information including DicerCall, Count, RPM, MajorRNA.
- The rowData includes the Cluster ID, the DicerCounts & the DicerConsensus.
- The colnames represents the sample replicate names.

For mRNAseq data

- The rownames contain the gene names.
- The assays represent the additional information including Count & FPKM.
- The rowData includes the gene & the SampleCounts.
- The colnames represents the sample replicate names.

Examples

```
# load data.frame
data("sRNA_data")

se <- RNAdf2se(input = "sRNA", data = sRNA_data)
```

RNAdicercall *Define the consensus dicercall for each sRNA cluster*

Description

The sRNA dicercall represents the length in nucleotides of the most abundant sRNA sequence within a cluster. The function calculates the consensus dicercall classification.

Usage

```
RNAdicercall(
  data,
  conditions = NULL,
  ties.method = NULL,
  tidy = FALSE,
  chimeric = FALSE,
  controls = NULL,
  genome.ID = NULL
)
```

Arguments

data	data.frame; originally generated by RNAimport().
conditions	character; vector containing sample replicate names. When supplied, the data from the named replicates will be the only ones used to calculate the dicercall consensus for each sRNA cluster. Each string should represent a sample name present in the dataframe supplied to the data argument.
ties.method	character; string specifying how ties are handled, choose either "exclude" or "random". When using random, if there is a tie one of the classes will be chosen at random. While, when using exclude if there is a tie the class is set to undefined, however, if there is a tie between a undefined and a known class, the known class takes president (eg 3x24-nt and 3xN-nt, then it will be classed as 24nt). Default setting ties.method="exclude".
tidy	logical; tidy-up data by removing sRNA clusters with an unknown or unclassified result. Default setting tidy=FALSE, removes excess background noise.
chimeric	logical; state whether the system is chimeric and contains multiple genomes/genotypes.
controls	character; vector of control condition sample names.
genome.ID	character; chromosome identifier of the genome representing either the origin of mobile molecules or the other genome in the chimeric system.

Details

For each sample, the alignment/clustering step predicted the sRNA dicercall for each cluster. This value is stored in the columns starting with "DicerCall_". This value represents the length of nucleotides of the most abundant sRNA within the cluster. For some clusters, there is no particular

sRNA which is more abundant than another, hence, it is stated as "NA" or "N", which is referred to as unclassified. The `RNAdicercall()` function calculate the consensus dicercall for each sRNA cluster based on the values across replicates. There are several parameters which will alter the output, including the handling of ties and the method to draw the consensus from.

When ties.method = "random", as per default, ties are broken at random. In this case, the determination of a tie assumes that the entries are probabilities: there is a relative tolerance of 1e-5, relative to the largest (in magnitude, omitting infinity) entry in the row.

When ties.method = "exclude", ties between sRNA classification are ruled as unclassified ("N"). However, when there is a tie between the choice of a class or unclassified result the `exclude` option will always select the class choice over the unclassified result.

If users are working with a chimeric system, utilise the `chimeric=TRUE` parameter and state `genome.ID` and `controls` parameter variables. This will remove any potential mapping errors which could introduce false interpretation.

To remove excess data noise, `tidy=TRUE` can be used to removed unclassified ("N") sRNA clusters, resulting in a reduced data set size.

Value

The original input data with two additional columns appended known as `DicerCounts` and `DicerConsensus`. The `DicerCounts` column stores the number of replicates that contributed to defining the consensus dicer-derived sRNA class. Note that when utilising the `exclude` ties methods, the `DicerCounts` will be represented as 0 when a tie is identified. While, the `DicerConsensus` stores the consensus dicercall.

Examples

```
 # load data
 data("sRNA_data")

# define consensus sRNA classes.
conditions <- c("heterograft_1", "heterograft_2", "heterograft_3")

# Run function to define sRNA class for each cluster.
sRNA_data_dicercall <- RNAdicercall(data = sRNA_data,
                                    conditions = conditions,
                                    tidy=TRUE)
```

`RNAdifferentialAnalysis`

Differential Analysis using `DESeq2` *or* `edgeR`

Description

`RNAdifferentialAnalysis` function computes the differential analysis with `DESeq2` or `edgeR` of sRNA or mRNA data produced by the `mobileRNA` package pipeline.

Usage

```
RNAdifferentialAnalysis(
  data,
  group,
  method = c("edgeR", "DESeq2"),
  dispersionValue = NULL
)
```

Arguments

`data`	data.frame; originally generated by `RNAimport()`.
`group`	character; the condition of each sample in the experimental design, formatted in the order of samples shown in the `data` object from left to right.
`method`	character; method to undertaken differential analysis, choose from methods of either DESeq2::DESeq or edgeR::edgeR. Must be stated as either "DESeq2" or "edgeR" in the function.
`dispersionValue`	numeric; value to represent the dispersion value for the edgeR::edgeR method. Recommended for analysis in experiments without biological replicates.

Details

The user has the flexibility to choose the method that best suits their data. In this function, the `DESeq2` method, calculates the differentials based on the normalized count data based on the size factors. Whereas the `edgeR` method, calculates normalization factors estimates dispersion, and returns the common dispersion. After normalization, the mean expression levels across samples are calculated, and differential expression analysis is performed using the exact test within groups, and the adjusted p-values are calculated using the Benjamini-Hochberg method. Note that this function is only capable of handling one replicate per condition with the `edgeR` method. This requires setting a suitable dispersion value. The dispersion value is other wise known as the common Biological squared coefficient of variation. See the User's Guide for the `edgeR` package for more details, edgeR::edgeR.

Value

Undertakes differential analysis, based on a specified method, and appends the results to the supplied data frame. This includes:

- Count mean
- Log fold change
- p-value
- Adjusted p-value
- Comparison order

Examples

```
# load data
data("sRNA_data")


# sample conditions.
groups <- c("Selfgraft", "Selfgraft", "Selfgraft", "Heterograft", "Heterograft", "Heterograft")


## Differential analysis: DEseq2 method
sRNA_DESeq2 <- RNAdifferentialAnalysis(data = sRNA_data,
                               group = groups,
                               method = "DESeq2" )


## Differential analysis: edgeR method
sRNA_edgeR <- RNAdifferentialAnalysis(data = sRNA_data,
                              group = groups,
                              method = "edgeR" )
```

RNAdistribution *Plot the distribution of sRNA classes based on nucleotide length*

Description

RNAdistribution plots the distribution of dicer-derived sRNA classes across samples or the sRNA consensus determined by the RNAdicercall() function. This can be displayed as a line or bar plot.

Usage

```
RNAdistribution(
  data,
  samples = NULL,
  style,
  data.type = "samples",
  facet = TRUE,
  facet.arrange = 3,
  colour = "#0868AC",
  outline = "black",
  wrap.scales = "fixed",
  overlap = TRUE,
  relative = FALSE,
  non.classified = TRUE
)
```

Arguments

data	data.frame; generated originally by RNAimport()
samples	character; states a subset of samples to plot. This argument is based on the sample names within the data.
style	character; style="line" or style="bar". Instructs how to plot the data. Where style="line" plots a line graph and style="bar" plots a bar graph.
data.type	character; either plotting "samples" or the "consensus" stored in the 'DicerConsensus' column.
facet	logical; forms a matrix of panels defined by row and column faceting variables. It plots the results for each sample as a bar chart and contains it within a single plot. The number of rows in the facet can be changed using the argument facet.arrange. Default facet = TRUE , plots each sample separately when facet = FALSE .
facet.arrange	numeric; value supplied to define the number of columns to include in the facet. This argument is piped into the ncol argument in ggplot2::facet_wrap() to define the number of columns. By default, this is set to 3.
colour	character; fill colour, Default is "#0868AC".
outline	character; states the outline colour for a box plot. Default is "black".
wrap.scales	character; scales be fixed ("fixed", the default), free ("free"), or free in one dimension ("free_x", "free_y")
overlap	logical; generates a single line graph, containing all sample replicate information. Default overlap=TRUE.
relative	logical; used in conjunction with data.type="consensus". Instructs plotting of the relative frequency of all sRNA classes across the data defined by the consensus dicer-derived sRNA column (see RNAdicercall() function for more information).
non.classified	logical; to include distribution of sRNAs which were unclassified. Default is TRUE, ie. maintain.

Details

The function can be used to plot a variety of different comparisons and plots. It can be used to plot the distribution of sRNA classes within each sample replicate, which can be represented as a bar chart style="bar" or a line graph style="line". These plots can be represented individually or in a single facet plot when facet="TRUE".

Additionally, there is the option to plot the distribution of sRNA classes within individual samples or to plot the distribution of the consensus dicer-derived sRNA classes determined by the RNAdicercall() function and stored in the column DicerConsensus when data.type="consensus". When plotting samples individually, there is the option to overlap the results onto a single line graph when overlap=TRUE. This is not an option for bar plots.

Value

The function returns a list containing the results: a data frame and the plot(s). To access an element, simply use the "$" symbol, and the elements "data" and "plot" will appear.

Examples

```
# load data
data('sRNA_data')

p1 <- RNAdistribution(data = sRNA_data, style = "line")

p2 <- RNAdistribution(data = sRNA_data, style = "line", overlap = FALSE)

p3 <- RNAdistribution(data = sRNA_data, style = "bar")

p3.2 <- RNAdistribution(data = sRNA_data, style = "bar",
                        samples = c("heterograft_1", "heterograft_2",
                        "heterograft_3"))

p4 <- RNAdistribution(data = sRNA_data, style = "bar", facet = FALSE)

p5 <- RNAdistribution(data = sRNA_data, style = "bar",
                      facet = TRUE, facet.arrange = 2 )


# Run function to define sRNA class for each cluster.
sRNA_data_dicercall <- mobileRNA::RNAdicercall(data = sRNA_data, tidy=TRUE)

p6 <- RNAdistribution(data = sRNA_data_dicercall, style = "bar", data.type = "consensus")
```

RNAfeatures *Summarise the distribution of sRNA clusters across genomic features*

Description

Calculates the number/proportion of genomic features within the supplied annotations and calculates the number/proportion of sRNA clusters which overlap with these genomic features.

Usage

```
RNAfeatures(
  data,
  annotation,
  repeats = NULL,
  repeat.type = NULL,
  promoterRegions = 1000,
  rmFeatures = NULL
)
```

Arguments

data	data.frame; generated by RNAimport()

annotation	path; URL or connection to a GFFFile object. A genome reference annotation file (.gff/.gff1/.gff2/.gff3). Can be in compressed format (gzip).
repeats	path; URL or connection to a GFFFile object. A genome reference annotation file, which only contains information on repeat sequences in the genome (.gff/.gff1/.gff2/.gff3). By default, this is not required, however if there is a specific repeats annotation file for the genome it is suggested to supply it. Can be in compressed format (gzip).
repeat.type	character; features type in annotation file to represent repeats when repeats not supplied.
promoterRegions	numeric; defines the upstream promoter region of genes. Default is 1000, which refers to promoters set at 1Kb upstream of genes
rmFeatures	vector; type of genomic features to not consider such as "gene" or "exon" etc.

Details

RNAfeatures calculates the number/percentage of sRNA clusters which overlap with genomic features based on their genomic coordinates.

Value

Returns a table containing the number and percentage of overlaps in the supplied sRNA data set with genomic features within supplied annotation and/or with repeats.

See Also

RNAmergeAnnotations() to merge 2 GFF files into 1.

Examples

```
data("sRNA_data")
features <- RNAfeatures(data = sRNA_data,
                        annotation = system.file("extdata",
                        "reduced_chr2_Tomato.gff.gz", package="mobileRNA"))
```

RNAimport *Import and organise sRNAseq & mRNAseq data sets*

Description

Load and organise either sRNAseq or mRNAseq pre-processing results into a single dataframe containing all experimental replicates specified where rows represent either a sRNA cluster (ie. sRNA producing-locus) or gene, respectively. Based on using the mobileRNA pre-processing method (See mapRNA()).

Usage

```
RNAimport(
  input = c("sRNA", "mRNA"),
  directory,
  samples,
  analysisType = "mobile",
  annotation,
  idattr = "Name",
  FPKM = FALSE,
  featuretype = "mRNA"
)
```

Arguments

input	string; define type of dataset. "sRNA" for sRNAseq data and "mRNA" for mRNAseq data.
directory	path; directory containing of sample folders generated by ShortStack
samples	character; vector naming samples correlating to outputted folders within the directory path.
analysisType	character; either "core" or "mobile" to represent the sRNA analysis workflow. Where the "core" sRNA analysis imports all reads (unique & multi-map), while "mobile" sRNA analysis imports only uniquely aligned read counts. Only for sRNA data, default is "mobile".
annotation	path; directory to genome annotation (GFF) file used for pre-processing. Only for mRNA data.
idattr	character; GFF attribute to be used as feature ID containing mRNA names. Several GFF lines with the same feature ID will be considered as parts of the same feature. The feature ID is used to identity the counts in the output table. Default is "Name". Only for mRNA data.
FPKM	logical; calculate the FPKM for each sample. Default is FALSE.
featuretype	character; type of feature. Default is "mRNA", only for mRNA data.

Details

The RNAimport() function requires the user to supply a directory path and a character vector. The path must be to the pre-processing output.

Following the mobileRNA method, for sRNA analysis, the path will be to the 2_alignment_results folder. While for mRNA analysis, the path will be to the 2_raw_counts folder. Both folders are generated by the mapRNA() function. The vector should contain strings that represent and mirror the names of the sample replicate folders in the above directory.

Together this information allows the function to extract the information stored in "Result.txt" files of each sample.

Value

For sRNAseq: A dataframe where rows represent sRNA clusters and columns represent replicate information extracted from the ShortStack output. Replicate information includes Dicercall, Counts, and MajorRNA sequence. Each replicate information is distinguishable as the replicate name is joined as a suffix to each column name. For example, for a sample called "Sample1", the columns will include DicerCall_Sample1, Count_Sample1, MajorRNA_Sample1 and RPM_Sample1.

The breakdown of each column:

- `Locus` : sRNA cluster locus
- `chr` : Chromosome
- `start` : start coordinate of cluster
- `end` : end coordinate of cluster
- `Cluster` : name of cluster
- `DicerCall_` : the size in nucleotides of most abundant sRNA in the cluster
- `Count_` : number of uniquely aligned sRNA-seq reads that overlap the locus
- `MajorRNA_` : RNA sequence of the most abundant sRNA in the cluster
- `RPM_` : reads per million
- `FPKM_` : Fragments Per Kilobase of transcript per Million mapped reads (only if option activated)

For mRNAseq: A dataframe where rows represent genes and columns represent replicate information extracted from HTseq result. Replicate information includes Counts and FPKM. For example, for a sample called "Sample1", the columns will include Count_Sample1, and FPKM_Sample1.

The breakdown of each column:

- `mRNA` : Name of mRNA
- `Locus`: Genomic loci of mRNA
- `chr` : Chromosome
- `start` : start coordinate
- `end` : end coordinate
- `width`: width in nucleotides of regions
- `Count_` : number of uniquely aligned mRNA-seq reads that overlap the locus
- `FPKM_` : Fragments Per Kilobase of transcript per Million mapped reads

References

ShortStack https://github.com/MikeAxtell/ShortStack, HISAT2 https://anaconda.org/bioconda/hisat2, HTSeq https://htseq.readthedocs.io/en/master/install.html, SAMtools https://anaconda.org/bioconda/samtools

Examples

```
## Not run:
# import sRNAseq data
df_sRNA <- RNAimport(input = "sRNA",
                     directory = "./analysis/sRNA_mapping_results",
                     samples = c("heterograft_1", "heterograft_2",
                     "heterograft_3","selfgraft_1" , "selfgraft_2" ,
                     "selfgraft_3"))


# The output of this function can be explored in the data object sRNA_data
data("sRNA_data")
head(sRNA_data)


# import sRNAseq data
df_mRNA <- RNAimport(input = "mRNA",
                     directory = "./analysis/mRNA_mapping_results",
                     samples = c("heterograft_1", "heterograft_2",
                     "heterograft_3","selfgraft_1" , "selfgraft_2" ,
                     "selfgraft_3"),
                     annotation = "./merged_annotation.gff3")



## End(Not run)
```

RNAmergeAnnotations *Merge two genome annotation files (GFF Format)*

Description

Merges two genomes annotation files (GFF) into one single GFF format file saved to the desired directory. This function adds a unique prefix to the chromosome names in each genome annotation to ensure each is distinguishable within the merged file.

Usage

```
RNAmergeAnnotations(
  annotationA,
  annotationB,
  output_file,
  AnnoA.ID = "A",
  AnnoB.ID = "B",
  format = "gff3"
)
```

Arguments

annotationA	path; path to a genome annotation assembly file in GFF format.
annotationB	path; path to a genome annotation assembly file in GFF format.
output_file	path; a character string or a base::connections() open for writing. Including file output name, and must have a GFF file extension.
AnnoA.ID	character; string to represent prefix added to existing chromosome names in annotationA. Default set as "A".
AnnoB.ID	character; string to represent prefix added to existing chromosome names in annotationB. Default set as "B".
format	format of GFF output, either "gff", "gff1", "gff2", "gff3." Default is "gff3".

Details

As default, the function removes periods and adds a prefix to the existing chromosome names. The prefix is separated from the original chromosome name by an underscore. For example, based on the default settings, it will add the prefix "A_" to the chromosome names in annotationA, for instance, A_0, A_1, A_2 etc.

The merged genome is saved to the specified output directory, and requires the user to set the name with a GFF format.

IMPORTANT: The genome reference and annotation of a species must have chromosomes with matching names. It is critical that if you used the RNAmergeGenomes() function to create a merged reference genome,that you treat the input annotations in the same way.

Value

A GFF format file containing the annotations of two genomes distinguishable by the appended prefixes.

Examples

```
anno1 <- system.file("extdata", "reduced_chr12_Eggplant.gff.gz",
package="mobileRNA")

anno2 <- system.file("extdata","reduced_chr2_Tomato.gff.gz",
package="mobileRNA")

output_file <- tempfile("merged_annotation", fileext = ".gff3")

merged_anno <- RNAmergeAnnotations(annotationA = anno1, annotationB = anno2,
                                   output_file = output_file)
```

`RNAmergeGenomes` *Merge two genome reference assemblies (FASTA format)*

Description

Merges two reference genomes (FASTA) into one single reference with modified chromosome names.

Typically, use genomeA as the origin tissue genome assembly, and genomeB as the genome from which mobile RNAs are produced by.

Usage

```
RNAmergeGenomes(
  genomeA,
  genomeB,
  output_file,
  GenomeA.ID = "A",
  GenomeB.ID = "B",
  compress.output = FALSE
)
```

Arguments

`genomeA`	path; path to a genome reference assembly file in FASTA format.
`genomeB`	path; path to a genome reference assembly file in FASTA format.
`output_file`	path; a character string or a `base::connections()` open for writing. Including file output name, and file extension of `.fa` or `.fasta`.
`GenomeA.ID`	character; string to represent prefix added to existing chromosome names in `genomeA`. Default set as "A"
`GenomeB.ID`	character; string to represent prefix added to existing chromosome names in `genomeB`. Default set as "B".
`compress.output`	logical; state whether the output file should be in a compressed format (gzip)

Details

The function merges two FASTA files, however, when merging genomic files it is critical that the two genomes are distinguishable by the chromosome names. As a default setting, the function extracts the chromosome names for the given FASTA files and alters adds a unique prefix while retaining the identifying number. Plus, removes any periods.

As default, the function will rename the chromosome names in `genome_A` to "A" and separates the prefix and the existing chromosome names with an underscore ("_"). For example, A_0, A_1, A_2 etc.

Please note that the underscore is added automatically, hence, when setting a custom prefix just includes character values.

IMPORTANT: The genome reference and annotation of the same species/accession/variety must have chromosomes with matching names. It is critical that if you use the RNAmergeAnnotations() function to create a merged genome annotation, that you treat the input references in the same way.

Value

Returns a single FASTA format file containing both genome assemblies with edited chromosome names (prefixes, and removal of periods) to the given directory.

Examples

```
fasta_1 <- system.file("extdata","reduced_chr12_Eggplant.fa.gz",
package="mobileRNA")

fasta_2 <-system.file("extdata","reduced_chr2_Tomato.fa.gz",
package="mobileRNA")


output_file <- file.path(tempfile("merged_annotation", fileext = ".fa"))

merged_ref <- RNAmergeGenomes(genomeA = fasta_1,
genomeB = fasta_2,
output_file = output_file)
```

RNAmobile *Identify putative RNA molecules produced by the non-tissue sample genome*

Description

A function to identify the putative sRNA or mRNA molecules produced by the non-tissue sample genome. Includes putative RNA mobilome or RNAs not expected to be found within the tissue of origin.

Usage

```
RNAmobile(
  input = c("sRNA", "mRNA"),
  data,
  controls,
  genome.ID,
  task = NULL,
  statistical = FALSE,
  alpha = 0.1,
  threshold = NULL
)
```

Arguments

input	character; must be either "sRNA" or "mRNA" to represent the type of data.
data	data.frame; generated through the **mobileRNA** method.
controls	character vector; containing names of control samples.
genome.ID	character; string or chromosome identifier related to the chromosomes in a given genome. A distinguishing feature of the genome of interest or non-interest in the chromosome name (chr column).
task	character; string to set the method to keep or remove the chromosomes containing the identifying string. To keep the chromosomes with the ID, set task=keep. To remove, set task="remove". As default, task is set to keep.
statistical	If TRUE, will undertake statistical filtering based on the a p-value or adjusted p-value threshold stated by alpha.Default set at FALSE. Requires presence of columns containing statistical data. In order to filter by the adjusted p-value, a column named padjusted must be present. See RNAdifferentialAnalysis() to calculate statistical values.
alpha	numeric; adjusted p-value cutoff as the target FDR for independent filtering. Default is 0.1. Only mobile molecules with adjusted p-values equal or lower than specified are returned.
threshold	numeric; set a threshold level. For sRNA analysis, this represents filtering by the minimum number of replicates that defined the consensus dicercall which is stored in the DicerCounts column. While, for mRNA analysis this represents the number of replicates which contained reads for the mRNA molecule which is stored in the SampleCounts column.

Details

The function identifies candidate sRNAs or mRNAs produced by a specific genome/genotype. It does so by either keeping or removing those mapped to a given genome. To do so, it requires a common pre-fix across chromosomes of the given genome. SeeRNAmergeGenomes() for more information. In addition, it removes RNAs which were likely to be falsely mapped. These are those which were mapped to the non-tissue genotype in the control samples.

For sRNAseq: A greater confidence in the sRNA candidates can be achieved by setting a threshold that considers the number of replicates which contributed to defining the consensus dicercall (ie. consensus sRNA classification). This parameter filters based on the DicerCounts column introduced by the RNAdicercall() function.

For mRNAseq: A greater confidence in the mRNA candidates can be achieved by setting a threshold that considers the number of replicates which contained reads for the mRNA molecule. This parameter filters based on the SampleCounts column introduced by the RNAimport() function.

Alternatively, utilize the baymobil algorithm, which identifies the mobile mRNAs using exact Bayesian inference based on SNP variant information. This option requires the user to input substantially more information. Please be aware this option requires significant processing time (so, maybe get a brew while this runs). Note that this option produces additional files, see Value section for more details.

Statistical Analysis The function also allows for filtering using statistical inference generated from the differential analysis of the total data set using the function RNAdifferentialAnalysis().

When `statistical=TRUE`, the feature is enabled and selects molecules that meet the adjusted p-value cutoff defined by `alpha`.

Value

A data frame containing candidate mobile sRNAs or mRNAs, which could have been further filtered based on statistical significance and the ability to by-pass the thresholds which determine the number of replicates that defined the consensus dicercall (sRNA) or contributed to reads counts (mRNA).

References

baymobil https://doi.org/10.21203/rs.3.rs-2520491/v1

Examples

```
data("sRNA_data")


# vector of control names
controls <- c("selfgraft_1", "selfgraft_2" , "selfgraft_3")

# Locate potentially mobile sRNA clusters associated to tomato, no
# statistical analysis
mobile_df1 <- RNAmobile(input = "sRNA", data =  sRNA_data,
controls = controls, genome.ID = "B_", task = "keep", statistical = FALSE)
```

`RNApopulation` *Identify gained/lost RNA populations between treatment and control conditions*

Description

Identify unique sRNA or mRNA populations within a set of samples, typically within the same condition, compared to the other samples in the analysis. This can be known as lost or gained populations due to a treatment.

Usage

```
RNApopulation(
  data,
  conditions,
  statistical = FALSE,
  alpha = 0.05,
  chimeric = FALSE,
  controls = NULL,
```

```
  genome.ID = NULL,
  dual = TRUE
)
```

Arguments

data	data.frame; generated by RNAimport()
conditions	character; containing names of samples within conditions to locate unique sRNA clusters.
statistical	logical; If TRUE, will undertake statistical filtering based on the adjusted p-value cutoff stated by alpha. Default setting statistical=FALSE. Requires presence of columns containing statistical data. To filter by the adjusted p-value, a column named padjusted must be present. See RNAdifferentialAnalysis() to calculate statistical values.
alpha	numeric; user-defined numeric value to represent the adjusted p-value threshold to define statistical significance. Defaults setting alpha=0.05. Returns sRNA clusters or mRNA with adjusted p-values equal or lower than the threshold value.
chimeric	logical; state whether system is chimeric containing multiple genomes/genotypes.
controls	character; vector of control condition sample names.
genome.ID	character; chromosome identifier of the genome representing either the origin of mobile molecules or the other genome in the chimeric system.
dual	logical; works in corporation when chimeric=TRUE and removes sRNA clusters mapped to the genome of mobile molecules.

Details

The function selects RNA which are unique to a given condition and absent in the samples within the other condition(s). For instance, a treatment might encourage the production of unique sRNAs which are not produced in the control samples. The function can also select the unique populations which show statistical significance, based on a adjusted p-value cutoff. This must have been calculated previously, see RNAdifferentialAnalysis() function.

If users are working with a chimeric system, utilise the chimeric=TRUE parameter and state genome.ID and controls parameter variables. This will remove any potential mapping errors which could lead to false interpretation.

Value

A subset of the supplied data and prints summary metric of results including:

- the total number of sRNA clusters or mRNA in the data set
- the number & percentage of unique sRNA clusters or mRNA to your condition
- the samples in the condition

Examples

```
data("sRNA_data")

# Select sRNA clusters only in the heterograft samples (ie. treatment)

heterograft_pop <- RNApopulation(data = sRNA_data,
                                 conditions = c("heterograft_1",
                                                "heterograft_2",
                                                "heterograft_3"))
```

`RNAreorder` *Reorder the data frame for differential analysis, ensuring control verse treatment comparison*

Description

Re-organise the working data frame, placing control samples before treatment samples. This ensures differential analysis comparison between controls and treatment are in the correct arrangement.

Usage

```
RNAreorder(data, controls)
```

Arguments

`data`	data.frame; generated by `RNAimport()`
`controls`	character; vector of control condition sample names.

Value

A re-ordered/re-organised working data frame with control samples after the 5 cluster information columns, and treatment sample columns after the control sample columns.

Examples

```
# load data
data("sRNA_data")

controlReps <- c("selfgraft_1", "selfgraft_2", "selfgraft_3")

reorder_df <- RNAreorder(data = sRNA_data, controls = controlReps)
```

RNAsequences *Extract RNA sequence from sRNA clusters*

Description

RNAsequences extrapolates the RNA sequence for sRNA clusters through two different methods utilising the RNA sequence of the most abundant transcript identified within each replicate. This can either be determined by extracting the consensus sequence across replicates or by comparing the sequences across replicates and selecting the most abundant. In this second method ties between sequences can be seen, hence, the user must decide whether a sequence is then chosen at random from the most abundant or will exclude any sequence determination.

The function also calculates the RNA & DNA complementary sequences, as well as stating the length/width of the sequence.

Usage

```
RNAsequences(
  data,
  original = FALSE,
  method = c("consensus", "set"),
  match.threshold = 1,
  duplicates = "random",
  tidy = FALSE
)
```

Arguments

data	data.frame; generated by RNAimport()
original	logical; output results to working date as additional columns (original=TRUE), or output as new data frame (original=FALSE). by default, FALSE
method	character; string to define method. Either "consensus" or "set". The "consensus" method identifies the consensus sequences across replicated based on the bioseq package method. Whereas the "set" is based on the fixed sequences calculated for each replicate and whether they are exact matches or not.
match.threshold	numeric; the minimum number of replicates required to share the sRNA sequence to count as a match. Default is 1. Only applicable to the "set" method.
duplicates	character; string to define how to deal with a tie, "random" as default. Options include "random" and "exclude". Only applicable to the "set" method.
tidy	logical; tidy-up data set by removing sRNA clusters with a unknown or unclassified consensus sRNA sequence result. By default, tidy=FALSE, while tidy=TRUE removes sRNA clusters with an undetermined consensus RNA sequence.

Details

The set method checks whether each sample in the data set shares the same major sRNA sequence for a given sRNA cluster. If at least two replicates share the same sRNA sequence, the sequence is pulled and the complementary DNA and RNA sequences are calculated. Using the `match.threshold` parameter, we can alter the minimum number of replicates required to share the RNA sequence to count as a match. For example, if set as `match.threshold=3`, at least 3 replicates must contain the same sequence. As a general rule, if only one replicate has determined a sRNA sequence it is noted that there is no match, but the sequence is pulled and the complementary sequences calculated.

The match column can either return "Yes", "No" or "Duplicate". If a match between replicates is found, "Yes" is supplied, if not, "No". While if there is a tie between sequences "Duplicate" is supplied. For examples, if an equal number of replicates have sequence "x" and sequence "y".

In the situation where duplicates are identified, as default, at random a consensus sRNA sequence is selected. This parameter can be changed to "exclude", and under this parameter no consensus sequence is pulled.

Whereas with the consensus method, the consensus sequence is pulled from all replicates.

Value

The results can be added as additional columns to the working data frame supplied to the function or stored as a new data frame containing only the results from the function. The results includes:

- Match: whether the RNA sequence is consistent across replicates
- Sequence: character; sequence of the most abundant sRNA within a cluster
- Complementary_RNA: character; complementary RNA nucleotide sequence
- Complementary_DNA: character; complementary DNA nucleotide sequence
- Width: numeric; length of nucleotide sequence

Examples

```
data("sRNA_data")

# vector of control names
controls <- c("selfgraft_1", "selfgraft_2" , "selfgraft_3")

# Locate potentially mobile sRNA clusters associated to tomato, no
# statistical analysis
sRNA_data_mobile <- RNAmobile(input = "sRNA", data =  sRNA_data,
controls = controls, genome.ID = "B_", task = "keep", statistical = FALSE)


mobile_sequences <- RNAsequences(sRNA_data_mobile, method = "consensus")
```

RNAsubset *Subset sRNA data based on dicercall size*

Description

Subset the existing dataframe to contain only the desired sRNA class(s) based on the consensus dicercall determination and statistical significance.

Usage

```
RNAsubset(data, class, statistical = FALSE, alpha = 0.05)
```

Arguments

data	data.frame; generated by RNAimport(), containing the additional information generated by RNAdicercall() which defines the consensus sRNA class for each sRNA dicer-derived clusters
class	numeric; sRNA dicercall class(es) to select. Based on size in nucleotides.
statistical	logical; filter and select statistically significant sRNA. If statistical=TRUE, sRNA clusters are selected based on p-adjusted alpha threshold.
alpha	numeric; user-defined numeric value to represent the adjusted p-value threshold to define statistical significance. Defaults setting alpha=0.05.

Details

See RNAdicercall() for information on defining the consensus sRNA dicercall class for each cluster. The function allows the choice to filtered the data by statistical significance based on differential expression analysis, see RNAdifferentialAnalysis(). Set statistical=TRUE to filtered by statistical significance (p-adjusted).

It is important to consider what point in your analysis the data is subset as it will drastically reduce your sample size, altering the output of statistical analyse.

Value

A data frame containing only sRNA clusters defined with a specific sRNA dicer-derived consensus size.

Examples

```
# load data
 data("sRNA_data")

# define consensus sRNA classes.
conditions <- c("heterograft_1", "heterograft_2", "heterograft_3")

# Run function to define sRNA class for each cluster.
sRNA_data_dicercall <- RNAdicercall(data = sRNA_data,
```

```
                                  conditions = conditions,
                                  tidy=TRUE)

# Subset data for  24-nt sRNAs
sRNA_24 <- RNAsubset(sRNA_data_dicercall, class = 24)


# Subset data for 21/22-nt sRNAs
sRNA_2122 <- RNAsubset(sRNA_data_dicercall, class = c(21, 22))
```

RNAsummary *Summarise differential analysis results*

Description

Print a summary of the statistical analysis of sRNA clusters (sRNAseq) or mRNAs (mRNAseq) from a mobileRNA analysis

Usage

```
RNAsummary(
  data,
  alpha = 0.1,
  chimeric = FALSE,
  controls = NULL,
  genome.ID = NULL
)
```

Arguments

`data`	data.frame; generated by `RNAdifferentialAnalysis()`
`alpha`	numeric; user-defined numeric value to represent the adjusted p-value threshold to define statistical significance. Defaults setting `alpha=0.1`.
`chimeric`	logical; state whether system is chimeric: contains multiple genomes/genotypes.
`controls`	character; vector of control condition sample names.
`genome.ID`	character; chromosome identifier of the genome representing either the origin of mobile molecules or the other genome in the chimeric system.

Details

To look only at the differential abundance from RNAs in the mobilome, use the `chimeric=TRUE` parameter and supply the chromosome identifier of the genome from which mobile molecules originate from to the `genome.ID` parameter & the control condition samples names to the `controls` parameter.

Value

Prints a summary of the RNAs which align with the adjusted p-value cutoff and states the number which has a positive and negative log-fold change. Where a positive log-fold change represents an increase in abundance and a negative value represents a decrease in abundance between the conditions.

Examples

```
# load data
data("sRNA_data")


# sample conditions.
groups <- c("Selfgraft", "Selfgraft", "Selfgraft", "Heterograft", "Heterograft", "Heterograft")


## Differential analysis: DEseq2 method
sRNA_DESeq2 <- RNAdifferentialAnalysis(data = sRNA_data,
                               group = groups,
                               method = "DESeq2" )


res <- RNAsummary(sRNA_DESeq2)
```

---

sRNA_data *sRNA_data: simulated data for biological replicates*

Description

Simulated sRNAseq dataset

Usage

```
data(sRNA_data)
```

Details

Simulates data is taken from eggplant and tomato sRNAseq samples and created to simulate to movement of sRNA molecules from an Tomato rootstock to an Eggplant Scion. Three Eggplant replicates were spiked with the same 150 tomato sRNA clusters, and named "heterograft_" 1 to 3. The analysis compares these heterografts to three Eggplant self-grafts which are the original un-spiked Eggplant replicates, called "selfgraft_" 1 to 3.

This data was imported and organised by the `RNAimport()` function.

Value

Dataframe in global environment

Examples

```
data("sRNA_data")
```

Index

∗ **internal**
    mobileRNA, 7

DESeq2::DESeq, *19*

edgeR::edgeR, *19*

ggplot2::facet_wrap(), *21*

mapRNA, 3, *7*
mapRNA(), *23, 24*
mobileRNA, 7
mobileRNA-package *(*mobileRNA*)*, 7
mRNA_data, 8

plotHeatmap, *8*, 9
plotRNAfeatures, 10
plotSampleDistance, *8*, 12
plotSamplePCA, *8*, 13

RNAattributes, *7*, 14
RNAdf2se, *8*, 16
RNAdicercall, *7*, 17
RNAdicercall(), *20, 21*, *30*, *36*
RNAdifferentialAnalysis, *7*, 18
RNAdifferentialAnalysis(), *30*, *32*, *36, 37*
RNAdistribution, *8*, 20
RNAfeatures, *8*, 22
RNAimport, *7*, 23
RNAimport(), *5, 6*, *8, 9*, *11–13*, *15*, *17*, *19*, *21, 22*, *30*, *32–34*, *36*, *38*
RNAmergeAnnotations, *7*, 26
RNAmergeAnnotations(), *11*, *15*, *23*, *29*
RNAmergeGenomes, *7*, 28
RNAmergeGenomes(), *15*, *27*, *30*
RNAmobile, *7*, 29
RNApopulation, *7*, 31
RNAreorder, *7*, 33
RNAsequences, *7*, 34
RNAsubset, *7*, 36
RNAsummary, *7*, 37

sRNA_data, 38