MJOLNIR3 Package Documentation

November 19, 2025

R topics documented:

`mjolnir1_RAN` *RAN: Demultiplex and trim primers from raw FASTQ files*

Description

This function will prepare the FASTQ raw data for each sample with the sample tags and primers removed. The R1 output files will contain all forward sequences and R2 the reverse sequences. Please, read the Details section before running.

Usage

```
mjolnir1_RAN(
  R1_filenames = "",
  lib_prefix = "",
  experiment = NULL,
  primer_F = "GGWACWRGWTGRACWNTNTAYCCYCC",
  primer_R = "TANACYTCNGGRTGNCCRAARAAYCA",
  cores = 1,
  R1_motif = "_R1",
  R2_motif = "_R2",
  metadata = "",
  multilane = FALSE,
  tag_error = 0,
  primer_error = 0.1,
  original_samples = "original_samples",
  mjolnir_agnomens = "mjolnir_agnomens",
  commands_file = "commands_runned_RAN.txt",
  only_commands = FALSE,
  ...
)
```

Arguments

`R1_filenames`	Character vector with the names of the forward fastq or fastq.gz files. Only needed for multiplexed libraries.
`lib_prefix`	Character vector. Acronym for each sequencing library. This acronym must be of 4 characters in capital letters. Do not mix up library and experiment acronyms. The latter will be required in following steps. However they can be the same.
`experiment`	Character string. Acronym for the experiment. This acronym must be of 4 characters in capital letters. Do not mix up library and experiment acronyms. However they can be the same.
`primer_F`	Character string of the Forward primer. Necessary when samples are already demultiplexed.
`primer_R`	Character string of the Reverse primer. Necessary when samples are already demultiplexed.
`cores`	Numeric. Number threads to run cutadapt.
`R1_motif`	Character string that distinguishes the forward line file from the reverse.
`R2_motif`	Character string that distinguishes the reverse line file from the forward.
`multilane`	Logical. If FALSE, the function will consider that the each library has only one sequencing lane. If TRUE, only one library is processed at a time.
`tag_error`	Numeric. From 0 (no errors allowed) to 1. will determine the proportion of nucleotides that can be different in the sample tags.
`primer_error`	Numeric. From 0 (no errors allowed) to 1. will determine the proportion of nucleotides that can be different in the primers.
`original_samples`	Character string. Name of the column in the metadata table that contains the original sample names.
`mjolnir_agnomens`	Character string. Name of the column in the metadata table that contains the mjolnir agnomen names.
`commands_file`	Character string. Name of the file where all commands will be written. If NULL or missing, commands will not be recorded.
`only_commands`	Logical. If TRUE, only the commands will be written to the commands_file. If FALSE, the commands will be executed.

`metadata_table`	tsv table. if not specified, the file must be named <EXPX>_metadata.tsv. This table must have: a column named "mjolnir_agnomens" with the names given to the samples during the pipeline in FREYJA; a column named "original_samples" with the samples names that will be given to the samples at the end of the pipeline; and a column with the name specified in in the "blank_col" parameter ("BLANK" by default) where blanks, negatives and controls are tagged with a flag specified in the "blank_tag" parameter (T by default).

Details

This function considers the following scenarios:

- **Scenario 1**. The samples are multiplexed in a library/libraries (it is possible to run multiple libraries at once from a single experiment). Each library has only two raw .fastq(.gz), R1 and R2. For each library there must be only one ngsfile (see below) and for the whole experiment only one metadata file (see below). mjolnir_agnomens (the standard name of the sample during the pipeline) must be unique, it must not be the same in different ngsfiles. *lib_prefix* and *R1_filenames* must be the same length.

- **Scenario 2**. The samples are multiplexed but for each library we have more than two raw data files. For example, when your library has been sequenced across multiple lanes on a Novaseq. In this case, this function has to be run separately for each library and the option *multilane* must be set to TRUE. In this case, *lib_prefix* can only be of length one (only one library), but the *R1_filenames* can be longer.

- **Scenario 3**. The samples are demultiplexed but the primer remains. In this case this function will trim the primers and separate the sequences into fwd and rev files.

Starting with multiplexed libraries:

Files required:

- *ngsfilter file*. Needed only for multiplexed libraries For each library, a ngsfilter file is needed and must be named **ngsfilter_<library identifier>.tsv**. This must contain five tab-separated columns and no header. The first column with the library identifier (four charachter identifier), the second with the mjolnir_agnomens, the third with the sample tags, the fourth with the forward primers and the fifth with the reverse primers.

- *metadata file*. A metadata file containing at least two columns required, "original_samples" and "mjolnir_agnomens" ("fastq_name_R1" for demultiplexed libraries, see below), and named as **<experiment identifier>_metadata.tsv**.

Important: when the same library has different sequencing lanes (NovaSeq), this function has to be run separately for each library and the option *multilane* must be set to TRUE.

Starting with demultiplexed samples:

If samples are already demultiplexed, primers need to be set (*primer_F* & *primer_R*) or LERAY_XT primers for COI will be used by default.

Files required:

- *metadata file*. This function will read the names of each individual R1 fastq files (full name including extension) from a column in the **<experiment identifier>_metadata.tsv** file, called "fastq_name_R1". In the metadata table, each sample in the "original_samples" column must have a their corresponding fastq_name_R1 and "mjolnir_agnomen" (LIBX_sample_XXX, i.e LIBA_sample_001). If the "fastq_name_R1" column is not in the metadata, This function will use the *lib_prefix*.

Examples

```
library(mjolnir)

# Define input fastq files (only names of R1 files are needed)
R1_filenames <- c("ULO1_R1.fastq.gz", "ULO2_R1.fastq.gz", "ULO3_R1.fastq.gz",
                  "ULO4_R1.fastq.gz")

# Input identifiers for the individual libraries to be used.
# It should be a 4-character name, matching the information in the
# ngsfilter files.
lib_prefix <- c("ULO1", "ULO2", "ULO3", "ULO4")

# experiment identifier
experiment <- 'ULOY'
# Enter number of cores to be used in parallel.
cores <- 7

mjolnir1_RAN(R1_filenames, lib_prefix = lib_prefix, experiment = experiment,
             cores = cores, R1_motif = "_R1", R2_motif = "_R2",
             tag_error = 0, primer_error = 0.1)
```

mjolnir2_FREYJA   *FREYJA: Filtering of Reads, Enrollment, Yoke-reads Joining and Alignment*

Description

FREYJA will use OBITools3 commands to merge paired-end reads, trim primer sequences, filter by length, split sequences per sample and dereplicate within each sample.

Usage

```
mjolnir2_FREYJA(
  experiment = NULL,
  cores = 1,
  Lmin = 299,
  Lmax = 320,
  min_overlap = 40,
  maxdiff = 0,
  error_rate = 0.5,
  R1_motif = "_R1",
  R2_motif = "_R2",
  commands_file = "commands_runned_FREYJA.txt",
  only_commands = FALSE,
  ...
)
```

Arguments

`experiment`	Character string. Acronym for the experiment. This acronym must be of 4 characters in capital letters. Do not mix up library and experiment acronyms. However they can be the same.
`cores`	Numeric. Number of threads for parallel processing.
`Lmin`	Numeric. Minimum bp length for a sequence to be accepted.
`Lmax`	Numeric. Maximum bp length for a sequence to be accepted.
`min_overlap`	Numeric. Minimum overlap length between R1 and R2 reads.
`maxdiff`	Numeric. Maximum number of differences allowed in the overlap. If less than 1, it will be considered as a percentage of the overlap length.
`error_rate`	Numeric. Maximum expected error rate allowed for the output merged reads. For a given sequence, the expected error is the sum of error probabilities for all the positions in the sequence. Since error probabilities can be small but not null, the expected error is always greater than zero, and at most equal to the length of the sequence when all positions in the sequence have an error probability of 1.0.
`R1_motif`	Character string that distinguish the forward line file from the reverse.
`R2_motif`	Character string that distinguish the reverse line file from the forward.
`commands_file`	Character string. Name of the file where all commands will be written. If NULL or missing, commands will not be recorded.
`only_commands`	Logical. If TRUE, only the commands will be written to the commands_file. If FALSE, the commands will be executed.

Details

Input file fastq files are expected to be without primers sequence and all forward sequences in the R1 file and all reverse sequences in the R2 file.

Examples

```
library(mjolnir)

# Define input fastq files (only names of R1 files are needed)
R1_filenames <- c("ULO1_R1.fastq.gz", "ULO2_R1.fastq.gz", "ULO3_R1.fastq.gz",
                  "ULO4_R1.fastq.gz")

# Input identifiers for the individual libraries to be used.
# It should be a 4-character name, matching the information in the
# ngsfilter files.
lib_prefixes <- c("ULO1", "ULO2", "ULO3", "ULO4")

# experiment identifier
experiment <- 'ULOY'
# Enter number of cores to be used in parallel.
cores <- 7

mjolnir1_RAN(R1_filenames, lib_prefix = lib_prefixes, experiment = experiment,
             cores = cores, R1_motif = "_R1", R2_motif = "_R2")
```

```
# Run FREYJA
mjolnir2_FREYJA(experiment = experiment, cores = cores, Lmin=299, Lmax=320)
```

---

mjolnir3_HELA *HELA: Hierarchical Elimination of Lurking Artifacts*

Description

This function uses the uchime_denovo algorithm implemented in VSEARCH to remove chimaeric sequences from the dataset.

Usage

```
mjolnir3_HELA(
  experiment = NULL,
  cores = 1,
  commands_file = "commands_runned_HELA.txt",
  only_commands = FALSE,
  ...
)
```

Arguments

`experiment`	Character string. Acronym for the experiment. This acronym must be of 4 characters in capital letters. Do not mix up library and experiment acronyms. However they can be the same.
`cores`	Numeric. Number of threads for parallel processing.
`commands_file`	Character string. Name of the file where all commands will be written. If NULL or missing, commands will not be recorded.
`only_commands`	Logical. If TRUE, only the commands will be written to the commands_file. If FALSE, the commands will be executed.

Details

HELA works in a sample-by-sample basis. HELA will process all individual fasta files in the current folder matching the pattern EXPX_XXXX_sample_XXX.fasta being EXPX the acronym set by experiment parameter. This allows for parallel computing, significantly decreasing calculation times.

Examples

```
library(mjolnir)

# Define input fastq files (only names of R1 files are needed)
R1_filenames <- c("ULO1_R1.fastq.gz", "ULO2_R1.fastq.gz", "ULO3_R1.fastq.gz",
                  "ULO4_R1.fastq.gz")
```

```
# Input identifiers for the individual libraries to be used.
# It should be a 4-character name, matching the information in the
# ngsfilter files.
lib_prefixes <- c("ULO1", "ULO2", "ULO3", "ULO4")

# experiment identifier
experiment <- 'ULOY'
# Enter number of cores to be used in parallel.
cores <- 7

mjolnir1_RAN(R1_filenames, lib_prefix = lib_prefixes, experiment = experiment,
             cores = cores, R1_motif = "_R1", R2_motif = "_R2")

# Run FREYJA
mjolnir2_FREYJA(experiment = experiment, cores = cores, Lmin=299, Lmax=320)

# Run HELA
mjolnir3_HELA(experiment = experiment, cores = cores)
```

mjolnir4_ODIN *ODIN: OTU Delimitation Inferred by Networks*

Description

ODIN performs MOTU clustering and/or denoising. It is one of the main steps of MJOLNIR.

ODIN performs MOTU clustering and/or denoising. It is one of the main steps of MJOLNIR.

Usage

```
mjolnir4_ODIN(
  experiment = NULL,
  cores = 1,
  d = 13,
  min_reads_MOTU = 2,
  min_reads_ESV = 2,
  min_relative = 1/50000,
  blank_relative = 0.1,
  metadata_table = "",
  blank_col = "BLANK",
  blank_tag = TRUE,
  alpha = 4,
  entropy = c(0.47, 0.23, 1.02, 313),
  algorithm = "DnoisE_SWARM",
  run_dnoise = TRUE,
  remove_singletons = NULL,
  ...
)
```

```
mjolnir4_ODIN(
  experiment = NULL,
  cores = 1,
  d = 13,
  min_reads_MOTU = 2,
  min_reads_ESV = 2,
  min_relative = 1/50000,
  blank_relative = 0.1,
  metadata_table = "",
  blank_col = "BLANK",
  blank_tag = TRUE,
  alpha = 4,
  entropy = c(0.47, 0.23, 1.02, 313),
  algorithm = "DnoisE_SWARM",
  run_dnoise = TRUE,
  remove_singletons = NULL,
  ...
)
```

Arguments

`experiment`	Character string. Acronym for the experiment. This acronym must be of 4 characters in capital letters. Do not mix up library and experiment acronyms. However they can be the same.
`cores`	Numeric. Number of threads for parallel processing.
`d`	Numeric value for d parameter of SWARM that refers to the maximum number of differences between two sequences to be linked in the same cluster
`min_reads_MOTU`	Numeric. Minimum number of reads that a MOTU needs to have to be retained.
`min_reads_ESV`	Numeric. Minimum number of reads that an ESV needs to have to be retained. Also works works for non-denoised sequences. In the case of algorithms involving SWARM, the removal is performed before SWARM.
`min_relative`	Number of the minimum relative abundance for a unit in the sample to be retained.
`blank_relative`	Relative abundance threshold for a unit to be removed if the total abundance in the blank/neg/control is higher than this value in terms of relative abundance of the total reads in all samples (see Details).
`metadata_table`	tsv table. if not specified, the file must be named <EXPX>_metadata.tsv. This table must have: a column named "mjolnir_agnomens" with the names given to the samples during the pipeline in FREYJA; a column named "original_samples" with the samples names that will be given to the samples at the end of the pipeline; and a column with the name specified in in the "blank_col" parameter ("BLANK" by default) where blanks, negatives and controls are tagged with a flag specified in the "blank_tag" parameter (T by default).
`blank_col`	Column name of the blank column in the "metadata_table"
`blank_tag`	Unique flag to tag the samples that are blank/neg/controls in the "blank_col"

`alpha`	Numeric. Alpha value for DnoisE to run.
`entropy`	Logical, numeric or character vector specifying whether to run DnoisE with entropy correction and how. a) c(0.47,0.23,1.02,313) - formulation refers to the entropy values for the first, second and third position in the codon and the length of the main sequence length expected. See Antich et al. 2022 for further details. b) FALSE - this will disable the entropy correction. Recommended for non coding markers c) c("auto_sample",313) - this will compute the entropy values for 313 (plus or minus multiple of 3) bp within DnoisE and use them to perform the entropy correction d) c("auto_dataset") - this will compute the entropy values for all the dataset and all sequence lengths and use the main sequence length's values for the entropy correction
`algorithm`	Character. It specifies the algorithm to obtain MOTUs and/or ESVs. Ther are four options: a)"DnoisE_SWARM" - This option will run DnoisE before SWARM. This option is the best choice for highly diverse data sets so it will reduce the computation time of SWARM. It also allows the analysis of metaphylogeographical approaches and retrieves denoised and quality filter fasta files for each sample that can be used for other experiment without previous run of RAN, FREYJA and HELA. It will result on a table of MOTUs and the ESV clustered into them. b)"SWARM" - This option will run only SWARM to obtain a MOTU table. c)"SWARM_DnoisE" - This option will run SWARM before DnoisE. This option responds to a philosophical point of view where the denoising of the sequences have to be performed within MOTUs so closer sequences are compared and to avoid that high abundant sequences from different MOTUs absorb sequences (and thus their reads) from different MOTUs. However, there are no major differences between this option and the "DnoisE_SWARM" option. It allows the analysis of metaphylogeographical and will result on a table of MOTUs and the ESV clustered into them. d)"DnoisE" - This option will run only DnoisE. This option will retrieve quality filter fasta files for each sample that can be used for other experiment without previous run of RAN, FREYJA and HELA.
`run_dnoise`	Logical. In the case of the algorithm='DnoisE_SWARM' there is the option of not running the DnoisE if it has been already run for a previous experiment. The denoised and filtered fasta files are needed.
`remove_DMS`	Logical. If TRUE, it will delete all obidms objects that are created during the process. This can save a lot of hard disk space. The FALSE option is useful for developing and debugging.

Details

The function mjolnir4_ODIN() uses the four different strategies to delimit MOTUs and/or ESVs. This strategies are set with the algorithm parameter:

a)"DnoisE_SWARM", b)"SWARM", c)"SWARM_DnoisE" and d)"DnoisE".

In short, DnoisE refers to the denoising process with DnoisE to obtain ESV and SWARM to a clustering process with SWARM to obtain MOTUs. DnoisE is a software to merge spurious sequences into their "mothers" (see Antich et al. 2022) to obtain Exact (also Amplicon) Sequence variants. DnoisE is an open source and parallelizable alternative to Unoise that allows to introduce an entropy correction based on the different entropies of each position in the codon of coding genes. This is highly recommended for markers as COI for which this program was intended. However, with the entropy=FALSE parameter, this programs performs the same denoising procedure as described for Unoise3. SWARM is an algorithm to delimit MOTUs, based on linkage-networks created by step-by-step agregation. This clustering algorithm is not based on an arbitrary, constant, absolute identity threshold. Conversely, SWARM is based on an iterative aggregation of sequences that differ less than a given distance d. This strategy results into linkage-networks of different sizes, which can have different effective values for within-MOTU identity threshold, depending on the complexity of the natural variability of the sequences present in the sample. This procedure is very convenient in the case of hypervariable metabarcoding markers, such as COI, which usually feature extremely high levels of natural diversity, in addition to the random sequencing errors. Dereplication step takes place after joining all samples into the same file before SWARM in algorithms a, b and c and after DnoisE in algorithms a and d

NEW: now ODIN performs most of the filters that were applied in RAGNAROC. The idea is to retrieve an output file for each sample that is independent from the rest of the samples (blank and neg corrected; total reads filtered) and with denoised ASV/ESV using the name

"<mjolnir_agnoment>_ODIN_ESV_<original_name>.fasta.".

To do so these are the the different steps of this function:

«D -> "DnoisE";

DS -> "DnoisE_SWARM"

S -> "SWARM";

SD -> "SWARM_DnoisE";

SaD (SWARM after DnoisE) –> "DnoisE_SWARM" & run_dnoise = F»

0: define variables and load metadata table

1: D and DS -> denoise the fasta files

2: D,DS,SD,S,SaD -> cat all fasta

3: D,DS,SD,S,SaD -> dereplicate

4: D,DS,SD,S,SaD -> annotate new names

filter1A: D,DS -> blank relative abundances filter AS MOTU

filter2A: D,DS -> min relative abundances filter WS ESV

filter3A: D,DS -> remove nletons # improves SWARM for DS

filter1B: SD,S,SaD -> remove nletons # this will improve SWARM performance # apply this to SaD just in case

5: D,DS,SD,S,SaD -> export csv file of sequences (if denoised, they are ESV)

6: D,DS -> export fasta DnoisEd & blank filt

7: D -> create fasta files for taxonomic assignment

8: DS,SD,S,SaD -> do SWARM

filter2B: SD,S -> blank relative abundances filter AS MOTU

filter3B: SD,S -> min relative abundances filter WS MOTU

filter4B: SD,S -> remove nletons AS MOTUs # remove artefacts

9: DS,SD,S,SaD -> create csv files of MOTUs and ESV or seqs.

10: DS,SD,S,SaD -> Also create fasta files for taxonomic assignment

11: SD -> run DnoisE over the csv files

Blank filter: remove any MOTU for which abundance in the blank or negative controls is higher than "blank_relative" of its total read abundance and remove blank and NEG samples

Minimum relative abundance filter: Apply a minimum relative abundance threshold for each sample, setting to zero any abundance below "min_relative" of the total reads of this sample. It also applies a "min_reads" filter

The function mjolnir4_ODIN() uses the four different strategies to delimit MOTUs and/or ESVs. This strategies are set with the algorithm parameter:

a)"DnoisE_SWARM", b)"SWARM", c)"SWARM_DnoisE" and d)"DnoisE".

In short, DnoisE refers to the denoising process with DnoisE to obtain ESV and SWARM to a clustering process with SWARM to obtain MOTUs. DnoisE is a software to merge spurious sequences into their "mothers" (see Antich et al. 2022) to obtain Exact (also Amplicon) Sequence variants. DnoisE is an open source and parallelizable alternative to Unoise that allows to introduce an entropy correction based on the different entropies of each position in the codon of coding genes. This is highly recommended for markers as COI for which this program was intended. However, with the entropy=FALSE parameter, this programs performs the same denoising procedure as described for Unoise3. SWARM is an algorithm to delimit MOTUs, based on linkage-networks created by step-by-step agregation. This clustering algorithm is not based on an arbitrary, constant, absolute identity threshold. Conversely, SWARM is based on an iterative aggregation of sequences that differ less than a given distance d. This strategy results into linkage-networks of different sizes, which can have different effective values for within-MOTU identity threshold, depending on the complexity of the natural variability of the sequences present in the sample. This procedure is very convenient in the case of hypervariable metabarcoding markers, such as COI, which usually feature extremely high levels of natural diversity, in addition to the random sequencing errors. Dereplication step takes place after joining all samples into the same file before SWARM in algorithms a, b and c and after DnoisE in algorithms a and d

NEW: now ODIN performs most of the filters that were applied in RAGNAROC. The idea is to retrieve an output file for each sample that is independent from the rest of the samples (blank and neg corrected; total reads filtered) and with denoised ASV/ESV using the name

"<mjolnir_agnoment>_ODIN_ESV_<original_name>.fasta.".

To do so these are the the different steps of this function:

«D -> "DnoisE";

DS -> "DnoisE_SWARM"

S -> "SWARM";

SD -> "SWARM_DnoisE";

SaD (SWARM after DnoisE) –> "DnoisE_SWARM" & run_dnoise = F»

0: define variables and load metadata table

1: D and DS -> denoise the fasta files

2: D,DS,SD,S,SaD -> cat all fasta

3: D,DS,SD,S,SaD -> dereplicate

4: D,DS,SD,S,SaD -> annotate new names

filter1A: D,DS -> blank relative abundances filter AS MOTU

filter2A: D,DS -> min relative abundances filter WS ESV

filter3A: D,DS -> remove nletons # improves SWARM for DS

filter1B: SD,S,SaD -> remove nletons # this will improve SWARM performance # apply this to SaD just in case

5: D,DS,SD,S,SaD -> export csv file of sequences (if denoised, they are ESV)

6: D,DS -> export fasta DnoisEd & blank filt

7: D -> create fasta files for taxonomic assignment

8: DS,SD,S,SaD -> do SWARM

filter2B: SD,S -> blank relative abundances filter AS MOTU

filter3B: SD,S -> min relative abundances filter WS MOTU

filter4B: SD,S -> remove nletons AS MOTUs # remove artefacts

9: DS,SD,S,SaD -> create csv files of MOTUs and ESV or seqs.

10: DS,SD,S,SaD -> Also create fasta files for taxonomic assignment

11: SD -> run DnoisE over the csv files

Blank filter: remove any MOTU for which abundance in the blank or negative controls is higher than "blank_relative" of its total read abundance and remove blank and NEG samples

Minimum relative abundance filter: Apply a minimum relative abundance threshold for each sample, setting to zero any abundance below "min_relative" of the total reads of this sample. It also applies a "min_reads" filter

Examples

```
library(mjolnir)

# Define input fastq files (only names of R1 files are needed)
R1_filenames <- c("ULO1_R1.fastq.gz", "ULO2_R1.fastq.gz", "ULO3_R1.fastq.gz",
                  "ULO4_R1.fastq.gz")

# Input identifiers for the individual libraries to be used.
# It should be a 4-character name, matching the information in the
# ngsfilter files.
lib_prefixes <- c("ULO1", "ULO2", "ULO3", "ULO4")

# experiment identifier
experiment <- 'ULOY'
# Enter number of cores to be used in parallel.
cores <- 7

mjolnir1_RAN(R1_filenames, lib_prefix = lib_prefixes, experiment = experiment,
             cores = cores, R1_motif = "_R1", R2_motif = "_R2")

# Run FREYJA
mjolnir2_FREYJA(experiment = experiment, cores = cores, Lmin=299, Lmax=320)

# Run HELA
mjolnir3_HELA(experiment = experiment, cores = cores)

# Run ODIN
mjolnir4_ODIN(experiment = experiment, cores = cores, d = 13,
              min_reads_MOTU = 2, min_reads_ESV = 2,
              min_relative = 1 / 50000, blank_relative = 0.1,
              metadata_table = "", blank_col = "BLANK", blank_tag = TRUE,
              alpha = 4, entropy = c(0.47, 0.23, 1.02, 313),
              algorithm = "DnoisE_SWARM")
library(mjolnir)

# Define input fastq files (only names of R1 files are needed)
R1_filenames <- c("ULO1_R1.fastq.gz", "ULO2_R1.fastq.gz", "ULO3_R1.fastq.gz",
                  "ULO4_R1.fastq.gz")

# Input identifiers for the individual libraries to be used.
# It should be a 4-character name, matching the information in the
# ngsfilter files.
lib_prefixes <- c("ULO1", "ULO2", "ULO3", "ULO4")

# experiment identifier
experiment <- 'ULOY'
# Enter number of cores to be used in parallel.
cores <- 7

mjolnir1_RAN(R1_filenames, lib_prefix = lib_prefixes, experiment = experiment,
             cores = cores, R1_motif = "_R1", R2_motif = "_R2")

# Run FREYJA
mjolnir2_FREYJA(experiment = experiment, cores = cores, Lmin=299, Lmax=320)

# Run HELA
mjolnir3_HELA(experiment = experiment, cores = cores)

# Run ODIN
mjolnir4_ODIN(experiment = experiment, cores = cores, d = 13,
              min_reads_MOTU = 2, min_reads_ESV = 2,
              min_relative = 1 / 50000, blank_relative = 0.1,
              metadata_table = "", blank_col = "BLANK", blank_tag = TRUE,
              alpha = 4, entropy = c(0.47, 0.23, 1.02, 313),
              algorithm = "DnoisE_SWARM")
```

mjolnir5_THOR *THOR: Taxonomy with Higher-than-Order Ranks*

Description

This is a wrapper of ecotag

Usage

```
mjolnir5_THOR(
  experiment = NULL,
  cores = 1,
  tax_db = NULL,
  run_ecotag = T,
  vsearch = F,
  remove_DMS = T,
  minimum_circle = 0.7,
  ...
)
```

Arguments

`experiment`	Character string. Acronym for the experiment. This acronym must be of 4 characters in capital letters. Do not mix up library and experiment acronyms. However they can be the same.
`cores`	Numeric. Number of threads for parallel processing.
`tax_db`	Character string specifying de PATH to the reference database. In case of using ecotag, the database must be an .obidms object. Also, when using ecotag it is important to have the following files in the same directory: order.complete.csv ; family_to_order.csv ; genus_to_family.csv
`run_ecotag`	Logical. Whether to run (TRUE, default) the ecotag taxonomic assignment or not (FALSE). The latter could take place when alternative taxonomic assignament software is applied but adding higher taxonomic ranks is desired.
`vsearch`	Logical. Whether to run (TRUE) the vsearch taxonomic assignment or not (FALSE, default). If vsearch has been selected, even if run_ecotag=TRUE, vsearch will be used instead of ecotag.
`remove_DMS`	Logical. If TRUE, it will delete all obidms objects that are created during the process. This can save a lot of hard disk space. The FALSE option is useful for developing and debugging.
`minimum_circle`	Numeric. For ecotag: Minimum identity considered for the assignment circle (sequence is assigned to the LCA of all sequences within a similarity circle of the best matches; the threshold for this circle is the highest value between <CIRCLE_THRESHOLD> and the best assignment score found for the query sequence). Give value as a normalized identity, e.g. 0.95 for an identity of 95 for a match. Default: 0.7

Details

After assignment with ecotag, higher taxa at ranks higher than order are added from cust. The database used can be download or build using the NJORDR package (see https://github.com/adriantich/NJORDR-MJOLNIR3)

For vsearch assignment, the database must be in fasta format and the taxonomy for the output CSV will be only in one single column.

Examples

```
library(mjolnir)

# Define input fastq files (only names of R1 files are needed)
R1_filenames <- c("ULO1_R1.fastq.gz", "ULO2_R1.fastq.gz", "ULO3_R1.fastq.gz",
                  "ULO4_R1.fastq.gz")

# Input identifiers for the individual libraries to be used.
# It should be a 4-character name, matching the information in the
# ngsfilter files.
lib_prefixes <- c("ULO1", "ULO2", "ULO3", "ULO4")

# experiment identifier
experiment <- 'ULOY'
# Enter number of cores to be used in parallel.
cores <- 7

mjolnir1_RAN(R1_filenames, lib_prefix = lib_prefixes, experiment = experiment,
             cores = cores, R1_motif = "_R1", R2_motif = "_R2")

# Run FREYJA
mjolnir2_FREYJA(experiment = experiment, cores = cores, Lmin=299, Lmax=320)

# Run HELA
mjolnir3_HELA(experiment = experiment, cores = cores)

# Run ODIN
mjolnir4_ODIN(experiment = experiment, cores = cores, d = 13,
              min_reads_MOTU = 2, min_reads_ESV = 2,
              min_relative = 1 / 50000, blank_relative = 0.1,
              metadata_table = "", blank_col = "BLANK", blank_tag = TRUE,
              alpha = 4, entropy = c(0.47, 0.23, 1.02, 313),
              algorithm = "DnoisE_SWARM")

# set the directory where the database is stored
tax_db <- "~/taxo_NCBI/DUFA_COI"

# Run THOR
mjolnir5_THOR(experiment = experiment, cores = cores,
              tax_db = tax_db, run_ecotag = T)
```

---

mjolnir6_FRIGGA *FRIGGA: Final Recount and Integration of Generated Genealogies and Abundances*

Description

FRIGGA recombined the abundances from ODIN with the taxonomic-annotated TSV files from THOR

Usage

```
mjolnir6_FRIGGA(experiment = NULL, ...)
```

Arguments

`experiment`	Character string. Acronym for the experiment. This acronym must be of 4 characters in capital letters. Do not mix up library and experiment acronyms. However they can be the same.

Details

Input files must be called <EXPX>_THOR_annotated.tsv for the taxonomic-annotated TSV file and <EXPX>_ODIN_counts.tsv for the read counts of MOTUs or <EXPX>_ODIN_counts.tsv for ESVs. Output file is then called <EXPX>_FRIGGA.tsv

Examples

```
library(mjolnir)

# Define input fastq files (only names of R1 files are needed)
R1_filenames <- c("ULO1_R1.fastq.gz", "ULO2_R1.fastq.gz", "ULO3_R1.fastq.gz",
                  "ULO4_R1.fastq.gz")

# Input identifiers for the individual libraries to be used.
# It should be a 4-character name, matching the information in the
# ngsfilter files.
lib_prefixes <- c("ULO1", "ULO2", "ULO3", "ULO4")

# experiment identifier
experiment <- 'ULOY'
# Enter number of cores to be used in parallel.
cores <- 7

mjolnir1_RAN(R1_filenames, lib_prefix = lib_prefixes, experiment = experiment,
             cores = cores, R1_motif = "_R1", R2_motif = "_R2")

# Run FREYJA
mjolnir2_FREYJA(experiment = experiment, cores = cores, Lmin=299, Lmax=320)

# Run HELA
mjolnir3_HELA(experiment = experiment, cores = cores)

# Run ODIN
mjolnir4_ODIN(experiment = experiment, cores = cores, d = 13,
              min_reads_MOTU = 2, min_reads_ESV = 2,
              min_relative = 1 / 50000, blank_relative = 0.1,
              metadata_table = "", blank_col = "BLANK", blank_tag = TRUE,
              alpha = 4, entropy = c(0.47, 0.23, 1.02, 313),
              algorithm = "DnoisE_SWARM")

# set the directory where the database is stored
tax_db <- "~/taxo_NCBI/DUFA_COI"

# Run THOR
mjolnir5_THOR(experiment = experiment, cores = cores,
              tax_db = tax_db, run_ecotag = T)

# Run FRIGGA
mjolnir6_FRIGGA(experiment = experiment)
```

mjolnir7_LOKI — *LOKI: LULU Overseeing with Kinship Identification*

Description

LOKI is a convenient wrapper of LULU for the MJOLNIR3 metabarcoding pipeline.

Usage

```
mjolnir7_LOKI(experiment = NULL, min_id = 0.84, discard = TRUE, ...)
```

Arguments

`experiment`	Character string. Acronym for the experiment. This acronym must be of 4 characters in capital letters. Do not mix up library and experiment acronyms. However they can be the same.
`min_id`	Numeric. Equivalent to the –id option from vsearch –usearch_global From vsearch manual: Reject the sequence match if the pairwise identity is lower than min_id (value ranging from 0.0 to 1.0 included). The search process sorts target sequences by decreasing number of k-mers they hav e in common with the query sequence, using that information as a proxy for sequence similarity. That efficient pre-filtering also prevents pairwise alignments with weakly matching targets, as there needs to be at least 6 shared kmers to start the pairwise alignment, and at least one out of every 16 k-mers from the query needs to match the target. Consequently, using values lower than –id 0.5 is not likely to capture more weakly matching targets. The pairwise identity is by default defined as the number of (matching columns) / (alignment length - terminal gaps)

Details

LOKI starts from the combined dataset of abundances and taxonomy from the previous step (FRIGGA): <EXPX>_FRIGGA.tsv. A match list of representative MOTU sequences is created using VSEARCH and saved as a txt file. Then Units that are potential errors based on co-occurrence patterns are labelled and removed using LULU. The output is called <EXPX>_LOKI_Curated.tsv

Examples

```
library(mjolnir)

# Define input fastq files (only names of R1 files are needed)
R1_filenames <- c("ULO1_R1.fastq.gz", "ULO2_R1.fastq.gz", "ULO3_R1.fastq.gz",
                  "ULO4_R1.fastq.gz")

# Input identifiers for the individual libraries to be used.
# It should be a 4-character name, matching the information in the
# ngsfilter files.
lib_prefixes <- c("ULO1", "ULO2", "ULO3", "ULO4")

# experiment identifier
experiment <- 'ULOY'
# Enter number of cores to be used in parallel.
cores <- 7

mjolnir1_RAN(R1_filenames, lib_prefix = lib_prefixes, experiment = experiment,
             cores = cores, R1_motif = "_R1", R2_motif = "_R2")

# Run FREYJA
mjolnir2_FREYJA(experiment = experiment, cores = cores, Lmin=299, Lmax=320)

# Run HELA
mjolnir3_HELA(experiment = experiment, cores = cores)

# Run ODIN
mjolnir4_ODIN(experiment = experiment, cores = cores, d = 13,
              min_reads_MOTU = 2, min_reads_ESV = 2,
              min_relative = 1 / 50000, blank_relative = 0.1,
              metadata_table = "", blank_col = "BLANK", blank_tag = TRUE,
              alpha = 4, entropy = c(0.47, 0.23, 1.02, 313),
              algorithm = "DnoisE_SWARM")

# set the directory where the database is stored
tax_db <- "~/taxo_NCBI/DUFA_COI"

# Run THOR
mjolnir5_THOR(experiment = experiment, cores = cores,
              tax_db = tax_db, run_ecotag = T)

# Run FRIGGA
mjolnir6_FRIGGA(experiment = experiment)

# Run LOKI
mjolnir7_LOKI(experiment = experiment, min_id = .84)
```

---

mjolnir8_RAGNAROC *RAGNAROC: Replace AGnomens with Names And Recover Original Codification*

Description

Final step of the MJOLNIR3 pipeline to apply the last filters to the abundance data

Usage

```
mjolnir8_RAGNAROC(
  experiment = NULL,
  metadata_table = "",
  output_file = "",
  output_file_ESV = "",
  min_reads = 0,
  remove_bacteria = T,
  remove_contamination = F,
  contamination_file = "contaminants.txt",
  ESV_within_MOTU = T,
  remove_numts = F,
  cores = 1,
  ...
)
```

Arguments

`experiment`	Character string. Acronym for the experiment. This acronym must be of 4 characters in capital letters. Do not mix up library and experiment acronyms. However they can be the same.
`metadata_table`	tsv table. if not specified, the file must be named <EXPX>_metadata.tsv. This table must have: a column named "mjolnir_agnomens" with the names given to the samples during the pipeline in FREYJA; a column named "original_samples" with the samples names that will be given to the samples at the end of the pipeline; and a column with the name specified in in the "blank_col" parameter ("BLANK" by default) where blanks, negatives and controls are tagged with a flag specified in the "blank_tag" parameter (T by default).
`output_file`	Character string specifying the outputfile name
`output_file_ESV`	Character string specifying the outputfile name for ESVs abundances if required.
`min_reads`	Number of the minimum number of reads allowed for each MOTU/ESV or ESV within MOTU.
`remove_bacteria`	Logical. If TRUE it will apply the bacteria removal filtering (see Details).
`remove_contamination`	Logical. If TRUE it will apply the contamination removal filtering (see Details).
`contamination_file`	Character string specifying the name of the contamination file. (see Details)
`ESV_within_MOTU`	Logical. If TRUE this will take into account the ESV that were clustered into MOTUs in ODIN if algorithm was set to "DnoisE_SWARM" or "SWARM_DnoisE" and apply all filters to both data.

`remove_numts`	Logical whether to apply the NUMT filter (TRUE) or not (FALSE)
`cores`	Numeric. Number of threads for parallel processing during NUMT removal.

Details

RAGNAROC consists on different contamination removals and filtering steps as follows:

Removal of Bacteria: this removed the Units tagged as "Prokaryota" or "root" in the <EXPX>_LOKI_Cutated.tsv

Removal of contaminations: this step removes the taxa specified in the "contaminaion_file"

NUMT removal: this step is design for the Leray-XT COI marker. It deletes all sequences that do not have a 313 (plus/minus multiple of 3 equivalent to a codon) bp length. Then removes sequences with stop codons and those metazoan sequences that do not translate for 5 specific conservative aminoacids.

Examples

```
library(mjolnir)

# Define input fastq files (only names of R1 files are needed)
R1_filenames <- c("ULO1_R1.fastq.gz", "ULO2_R1.fastq.gz", "ULO3_R1.fastq.gz",
                  "ULO4_R1.fastq.gz")

# Input identifiers for the individual libraries to be used.
# It should be a 4-character name, matching the information in the
# ngsfilter files.
lib_prefixes <- c("ULO1", "ULO2", "ULO3", "ULO4")

# experiment identifier
experiment <- 'ULOY'
# Enter number of cores to be used in parallel.
cores <- 7

mjolnir1_RAN(R1_filenames, lib_prefix = lib_prefixes, experiment = experiment,
             cores = cores, R1_motif = "_R1", R2_motif = "_R2")

# Run FREYJA
mjolnir2_FREYJA(experiment = experiment, cores = cores, Lmin=299, Lmax=320)

# Run HELA
mjolnir3_HELA(experiment = experiment, cores = cores)

# Run ODIN
mjolnir4_ODIN(experiment = experiment, cores = cores, d = 13,
              min_reads_MOTU = 2, min_reads_ESV = 2,
              min_relative = 1 / 50000, blank_relative = 0.1,
              metadata_table = "", blank_col = "BLANK", blank_tag = TRUE,
              alpha = 4, entropy = c(0.47, 0.23, 1.02, 313),
              algorithm = "DnoisE_SWARM")

# set the directory where the database is stored
tax_db <- "~/taxo_NCBI/DUFA_COI"
```

```
# Run THOR
mjolnir5_THOR(experiment = experiment, cores = cores,
              tax_db = tax_db, run_ecotag = T)

# Run FRIGGA
mjolnir6_FRIGGA(experiment = experiment)

# Run LOKI
mjolnir7_LOKI(experiment = experiment, min_id = .84)

# Run RAGNAROC
mjolnir8_RAGNAROC(experiment = experiment, remove_bacteria = T, ESV_within_MOTU = T,
                  remove_numts = F, cores = 1)
```

`numts` *numts*

Description

Internal function of RAGNAROC to detect numt ESV within MOTUs

Usage

```
numts(datas, is_metazoa = FALSE, motu, datas_length)
```

Arguments

`is_metazoa`	Logical. whether if the MOTU is assigned to metazoa or not
`motu`	Character. Name of the MOTU evaluated
`datas_length`	Numeric vector. Vector with the number of characters in each sequence
`data`	Data Frame of the sequences in the MOTU.

Details

numt function will: 1: keep sequences of the same length as the seed 2: remove missaligned sequences 3: search for codon stops using all the mitochondrial genome codes with the Biostrings package and keep the one with less reads with stop codons. 4: if the MOTU is assigned to metazoans and the seed has 313 bp then check it the following aminoacid positions have the conserved ones in metazoans: 20 = "H"; 23 ="G"; 32 = "N"; 81 ="D"; 95 ="G"

Index