Package 'dada2'

July 21, 2018

Type Package

Title Accurate, high-resolution sample inference from amplicon sequencing data

Description The dada2 package infers exact amplicon sequence variants (ASVs) from high-throughput amplicon sequencing data, replacing the coarser and less accurate OTU clustering approach. The dada2 pipeline takes as input demultiplexed fastq files, and outputs the sequence variants and their sample-wise abundances after removing substitution and chimera errors. Taxonomic classification is available via a native implementation of the RDP naive Bayesian classifier, and genus-species assignment by exact matching.

Version 1.8.0

Date 2018-1-30

Maintainer Benjamin Callahan <benjamin.j.callahan@gmail.com>

Author Benjamin Callahan <benjamin.j.callahan@gmail.com>, Paul McMurdie, Susan Holmes

License LGPL-3

LazyLoad yes

Depends R (>= 3.2.0), Rcpp (>= 0.11.2), methods (>= 3.2.0)

Imports Biostrings (>= 2.42.1), ggplot2 (>= 2.1.0), data.table (>= 1.9.4), reshape2 (>= 1.4.1), ShortRead (>= 1.32.0), RcppParallel (>= 4.3.0), parallel (>= 3.2.0), IRanges (>= 2.6.0), XVector (>= 0.16.0), BiocGenerics (>= 0.22.0)

Suggests BiocStyle, knitr, rmarkdown

LinkingTo Rcpp, RcppParallel

SystemRequirements GNU make

VignetteBuilder knitr

biocViews Microbiome, Sequencing, Classification, Metagenomics

URL http://benjjneb.github.io/dada2/

BugReports https://github.com/benjjneb/dada2/issues

LazyData true

Collate 'RcppExports.R' 'allClasses.R' 'allPackage.R' 'chimeras.R' 'dada.R' 'errorModels.R' 'filter.R' 'misc.R' 'multiSample.R' 'paired.R' 'plot-methods.R' 'sequenceIO.R' 'show-methods.R' 'taxonomy.R'

RoxygenNote 6.0.1

git_url https://git.bioconductor.org/packages/dada2

git_branch RELEASE_3_7

git_last_commit 630ef9a

git_last_commit_date 2018-04-30

Date/Publication 2018-07-21

R topics documented:

`dada2-package` *DADA2 package*

Description

The dada2 package is centered around the DADA2 algorithm for accurate high-resolution of sample composition from amplicon sequencing data. The DADA2 algorithm is both more sensitive and more specific than commonly used OTU methods, and resolves amplicon sequence variants (ASVs) that differ by as little as one nucleotide.

Details

The dada2 package also provides a full set of tools for taking raw amplicon sequencing data all the way through to a feature table representing sample composition. Provided facilities include:

- Quality filtering (`filterAndTrim`, `fastqFilter`, `fastqPairedFilter`)
- Dereplication (`derepFastq`)
- Learn error rates (`learnErrors`)
- Sample Inference (`dada`)
- Chimera Removal (`removeBimeraDenovo`, `isBimeraDenovo`, `isBimeraDenovoTable`)
- Merging of Paired Reads (`mergePairs`)
- Taxonomic Classification (`assignTaxonomy`, `assignSpecies`)

Author(s)

Benjamin Callahan <benjamin.j.callahan@gmail.com>

Paul J McMurdie II <mcmurdie@stanford.edu>

Michael Rosen <eigenrosen@gmail.com>

Susan Holmes <susan@stat.stanford.edu>

addSpecies *Add species-level annotation to a taxonomic table.*

Description

addSpecies wraps the assignSpecies function to assign genus-species binomials to the input sequences by exact matching against a reference fasta. Those binomials are then merged with the input taxonomic table with species annotations appended as an additional column to the input table. Only species identifications where the genera in the input table and the binomial classification are consistent are included in the return table.

Usage

```
addSpecies(taxtab, refFasta, allowMultiple = FALSE, tryRC = FALSE,
  n = 2000, verbose = FALSE)
```

Arguments

taxtab	(Required). A taxonomic table, the output of assignTaxonomy.
refFasta	(Required). The path to the reference fasta file, or an R connection. Can be compressed. This reference fasta file should be formatted so that the id lines correspond to the genus-species binomial of the associated sequence: >SeqID genus species ACGAATGTGAAGTAA.....
allowMultiple	(Optional). Default FALSE. Defines the behavior when multiple exact matches against different species are returned. By default only unambiguous identifications are return. If TRUE, a concatenated string of all exactly matched species is returned. If an integer is provided, multiple identifications up to that many are returned as a concatenated string.
tryRC	(Optional). Default FALSE. If TRUE, the reverse-complement of each sequences will be used for classification if it is a better match to the reference sequences than the forward sequence.
n	(Optional). Default 1e5. The number of records (reads) to read in and filter at any one time. This controls the peak memory requirement so that very large fastq files are supported. See FastqStreamer for details.
verbose	(Optional). Default FALSE. If TRUE, print status to standard output.

Value

A character matrix one column larger than input. Rows correspond to sequences, and columns to the taxonomic levels. NA indicates that the sequence was not classified at that level.

See Also

assignTaxonomy, assignSpecies

Examples

```
seqs <- getSequences(system.file("extdata", "example_seqs.fa", package="dada2"))
training_fasta <- system.file("extdata", "example_train_set.fa.gz", package="dada2")
taxa <- assignTaxonomy(seqs, training_fasta)
species_fasta <- system.file("extdata", "example_species_assignment.fa.gz", package="dada2")
taxa.spec <- addSpecies(taxa, species_fasta)
taxa.spec.multi <- addSpecies(taxa, species_fasta, allowMultiple=TRUE)
```

---

assignSpecies *Taxonomic assignment to the species level by exact matching.*

Description

`assignSpecies` uses exact matching against a reference fasta to identify the genus-species binomial classification of the input sequences.

Usage

```
assignSpecies(seqs, refFasta, allowMultiple = FALSE, tryRC = FALSE,
  n = 2000, verbose = FALSE)
```

Arguments

`seqs`	(Required). A character vector of the sequences to be assigned, or an object coercible by `getUniques`.
`refFasta`	(Required). The path to the reference fasta file, or an R connection. Can be compressed. This reference fasta file should be formatted so that the id lines correspond to the genus-species of the associated sequence: >SeqID genus species ACGAATGTGAAGTAA.....
`allowMultiple`	(Optional). Default FALSE. Defines the behavior when multiple exact matches against different species are returned. By default only unambiguous identifications are return. If TRUE, a concatenated string of all exactly matched species is returned. If an integer is provided, multiple identifications up to that many are returned as a concatenated string.
`tryRC`	(Optional). Default FALSE. If TRUE, the reverse-complement of each sequences will also be tested for exact matching to the reference sequences.
`n`	(Optional). Default `2000`. The number of sequences to perform assignment on at one time. This controls the peak memory requirement so that large numbers of sequences are supported.
`verbose`	(Optional). Default FALSE. If TRUE, print status to standard output.

Value

A two-column character matrix. Rows correspond to the provided sequences, columns to the genus and species taxonomic levels. NA indicates that the sequence was not classified at that level.

Examples

```
seqs <- getSequences(system.file("extdata", "example_seqs.fa", package="dada2"))
species_fasta <- system.file("extdata", "example_species_assignment.fa.gz", package="dada2")
spec <- assignSpecies(seqs, species_fasta)
```

`assignTaxonomy` *Classifies sequences against reference training dataset.*

Description

assignTaxonomy implements the RDP Naive Bayesian Classifier algorithm described in Wang et al. Applied and Environmental Microbiology 2007, with kmer size 8 and 100 bootstrap replicates. Properly formatted reference files for several popular taxonomic databases are available http://benjjneb.github.io/dada2/training.html

Usage

```
assignTaxonomy(seqs, refFasta, minBoot = 50, tryRC = FALSE,
  outputBootstraps = FALSE, taxLevels = c("Kingdom", "Phylum", "Class",
  "Order", "Family", "Genus", "Species"), multithread = FALSE,
  verbose = FALSE)
```

Arguments

`seqs`	(Required). A character vector of the sequences to be assigned, or an object coercible by `getUniques`.
`refFasta`	(Required). The path to the reference fasta file, or an R connection Can be compressed. This reference fasta file should be formatted so that the id lines correspond to the taxonomy (or classification) of the associated sequence, and each taxonomic level is separated by a semicolon. Eg. >Kingom;Phylum;Class;Order;Family;Genus; ACGAATGTGAAGTAA.....
`minBoot`	(Optional). Default 50. The minimum bootstrap confidence for assigning a taxonomic level.
`tryRC`	(Optional). Default FALSE. If TRUE, the reverse-complement of each sequences will be used for classification if it is a better match to the reference sequences than the forward sequence.
`outputBootstraps`	(Optional). Default FALSE. If TRUE, bootstrap values will be retained in an integer matrix. A named list containing the assigned taxonomies (named "taxa") and the bootstrap values (named "boot") will be returned. Minimum bootstrap confidence filtering still takes place, to see full taxonomy set minBoot=0
`taxLevels`	(Optional). Default is c("Kingdom", "Phylum", "Class", "Order", "Family", "Genus", "Species"). The taxonomic levels being assigned. Truncates if deeper levels not present in training fasta.
`multithread`	(Optional). Default is FALSE. If TRUE, multithreading is enabled and the number of available threads is automatically determined. If an integer is provided, the number of threads to use is set by passing the argument on to `setThreadOptions`.
`verbose`	(Optional). Default FALSE. If TRUE, print status to standard output.

Value

A character matrix of assigned taxonomies exceeding the minBoot level of bootstrapping confidence. Rows correspond to the provided sequences, columns to the taxonomic levels. NA indicates that the sequence was not consistently classified at that level at the minBoot threshhold.

If outputBootstraps is TRUE, a named list containing the assigned taxonomies (named "taxa") and the bootstrap values (named "boot") will be returned.

Examples

```
seqs <- getSequences(system.file("extdata", "example_seqs.fa", package="dada2"))
training_fasta <- system.file("extdata", "example_train_set.fa.gz", package="dada2")
taxa <- assignTaxonomy(seqs, training_fasta)
taxa80 <- assignTaxonomy(seqs, training_fasta, minBoot=80, multithread=2)
```

---

c,dada-method *Change concatenation to list construction.*

Description

Change concatenation to list construction.

Usage

```
## S4 method for signature 'dada'
c(x, ..., recursive = FALSE)
```

Arguments

x	A dada-class object
...	objects to be concatenated.
recursive	logical. If `recursive = TRUE`, the function recursively descends through lists (and pairlists) combining all their elements into a vector.

Value

list.

c,derep-method *Change concatenation to list construction.*

Description

Change concatenation to list construction.

Usage

```
## S4 method for signature 'derep'
c(x, ..., recursive = FALSE)
```

Arguments

x	A derep-class object
...	objects to be concatenated.
recursive	logical. If recursive = TRUE, the function recursively descends through lists (and pairlists) combining all their elements into a vector.

Value

list.

collapseNoMismatch *Combine together sequences that are identical up to shifts and/or length.*

Description

This function takes as input a sequence table and returns a sequence table in which any sequences that are identical up to shifts or length variation, i.e. that have no mismatches or internal indels when aligned, are collapsed together. The most abundant sequence is chosen as the representative of the collapsed sequences. This function can be thought of as implementing greedy 100% OTU clustering, with end-gapping is ignored.

Usage

```
collapseNoMismatch(seqtab, minOverlap = 20, orderBy = "abundance",
  vec = TRUE, verbose = FALSE)
```

Arguments

seqtab	(Required). A sample by sequence matrix, the return of makeSequenceTable.
minOverlap	(Optional). numeric(1). Default 20. The minimum amount of overlap between sequences required to collapse them together.
orderBy	(Optional). character(1). Default "abundance". Specifies how the sequences (columns) of the returned table should be ordered (decreasing). Valid values: "abundance", "nsamples", NULL.

vec (Optional). `logical(1)`. Default TRUE. Use the vectorized aligner. Should be turned off if sequences exceed 2kb in length.

verbose (Optional). `logical(1)`. Default FALSE. If TRUE, a summary of the function results are printed to standard output.

Value

Named integer matrix. A row for each sample, and a column for each collapsed sequence across all the samples. Note that the columns are named by the sequence which can make display a little unwieldy. Columns are in the same order (modulo the removed columns) as in the input matrix.

See Also

`makeSequenceTable`

Examples

```
derep1 <- derepFastq(system.file("extdata", "sam1F.fastq.gz", package="dada2"))
derep2 <- derepFastq(system.file("extdata", "sam2F.fastq.gz", package="dada2"))
dada1 <- dada(derep1, tperr1)
dada2 <- dada(derep2, tperr1)
seqtab <- makeSequenceTable(list(sample1=dada1, sample2=dada2))
collapseNoMismatch(seqtab)
```

---

dada *High resolution sample inference from amplicon data.*

Description

The dada function takes as input dereplicated amplicon sequencing reads and returns the inferred composition of the sample (or samples). Put another way, dada removes all sequencing errors to reveal the members of the sequenced community.

If dada is run in selfConsist=TRUE mode, the algorithm will infer both the sample composition and the parameters of its error model from the data.

Usage

```
dada(derep, err, errorEstimationFunction = loessErrfun, selfConsist = FALSE,
  pool = FALSE, priors = character(0), multithread = FALSE,
  verbose = FALSE, ...)
```

Arguments

derep (Required). A `derep-class` object, the output of `derepFastq`. A list of such objects can be provided, in which case each will be denoised with a shared error model.

err (Required). 16xN numeric matrix, or an object coercible by `getErrors` such as the output of the `learnErrors` function.

The matrix of estimated rates for each possible nucleotide transition (from sample nucleotide to read nucleotide). Rows correspond to the 16 possible transitions (t_ij) indexed such that 1:A->A, 2:A->C, ..., 16:T->T Columns correspond to quality scores. Each entry must be between 0 and 1.

Typically there are 41 columns for the quality scores 0-40. However, if USE_QUALS=FALSE, the matrix must have only one column.

If selfConsist = TRUE, `err` can be set to NULL and an initial error matrix will be estimated from the data by assuming that all reads are errors away from one true sequence.

`errorEstimationFunction`

(Optional). Function. Default `loessErrfun`.

If USE_QUALS = TRUE, `errorEstimationFunction(dada()$trans_out)` is computed after sample inference, and the return value is used as the new estimate of the err matrix in $err_out.

If USE_QUALS = FALSE, this argument is ignored, and transition rates are estimated by maximum likelihood (t_ij = n_ij/n_i).

`selfConsist` (Optional). `logical(1)`. Default FALSE.

If selfConsist = TRUE, the algorithm will alternate between sample inference and error rate estimation until convergence. Error rate estimation is performed by `errorEstimationFunction`.

If selfConsist=FALSE the algorithm performs one round of sample inference based on the provided `err` matrix.

`pool` (Optional). `logical(1)`. Default is FALSE.

If pool = TRUE, the algorithm will pool together all samples prior to sample inference. If pool = FALSE, sample inference is performed on each sample individually. If pool = "pseudo", the algorithm will perform pseudo-pooling between individually processed samples.

This argument has no effect if only 1 sample is provided, and `pool` does not affect error rates, which are always estimated from pooled observations across samples.

`priors` (Optional). `character`. Default is character(0), i.e. no prior sequences.

The priors argument provides a set of sequences for which there is prior information suggesting they may truly exist, i.e. are not errors. The abundance p-value of dereplicated sequences that exactly match one of the priors are calculated without conditioning on presence, allowing singletons to be detected, and are compared to a reduced threshold 'OMEGA_P' when forming new partitions.

`multithread` (Optional). Default is FALSE. If TRUE, multithreading is enabled and the number of available threads is automatically determined. If an integer is provided, the number of threads to use is set by passing the argument on to `setThreadOptions`.

`verbose` (Optional). Default FALSE. Print verbose text output. More fine-grained control is available by providing an integer argument.

- 0: Silence. No text output
- 1: Minimal text output (same as FALSE).
- 2: Detailed output (same as TRUE).

`...` (Optional). All dada_opts can be passed in as arguments to the dada() function. See `setDadaOpt` for a full list and description of these options.

Details

Briefly, dada implements a statistical test for the notion that a specific sequence was seen too many times to have been caused by amplicon errors from currently inferred sample sequences. Overly-abundant sequences are used as the seeds of new partitions of sequencing reads, and the final set of partitions is taken to represent the denoised composition of the sample. A more detailed explanation of the algorithm is found in two publications:

- Callahan BJ, McMurdie PJ, Rosen MJ, Han AW, Johnson AJ, Holmes SP (2016). DADA2: High resolution sample inference from Illumina amplicon data. Nature Methods, 13(7), 581-3.
- Rosen MJ, Callahan BJ, Fisher DS, Holmes SP (2012). Denoising PCR-amplified metagenome data. BMC bioinformatics, 13(1), 283.

dada depends on a parametric error model of substitutions. Thus the quality of its sample inference is affected by the accuracy of the estimated error rates. `selfConsist` mode allows these error rates to be inferred from the data.

All comparisons between sequences performed by dada depend on pairwise alignments. This step is the most computationally intensive part of the algorithm, and two alignment heuristics have been implemented for speed: A kmer-distance screen and banded Needleman-Wunsch alignmemt. See `setDadaOpt`.

Value

A `dada-class` object or list of such objects if a list of dereps was provided.

See Also

`derepFastq`, `setDadaOpt`

Examples

```
derep1 = derepFastq(system.file("extdata", "sam1F.fastq.gz", package="dada2"))
derep2 = derepFastq(system.file("extdata", "sam2F.fastq.gz", package="dada2"))
dada(derep1, err=tperr1)
dada(list(sam1=derep1, sam2=derep2), err=tperr1, selfConsist=TRUE)
dada(derep1, err=inflateErr(tperr1,3), BAND_SIZE=32, OMEGA_A=1e-20)
```

---

dada-class *The object class returned by* dada

Description

A multi-item List with the following named values...

- $denoised: Integer vector, named by sequence valued by abundance, of the denoised sequences.
- $clustering: An informative data.frame containing information on each cluster.
- $sequence: A character vector of each denoised sequence. Identical to names($denoised).
- $quality: The average quality scores for each cluster (row) by position (col).

- $map: Integer vector that maps the unique (index of derep$unique) to the denoised sequence (index of dada$denoised).
- $birth_subs: A data.frame containing the substitutions at the birth of each new cluster.
- $trans: The matrix of transitions by type (row), eg. A2A, A2C..., and quality score (col) observed in the final output of the dada algorithm.
- $err_in: The err matrix used for this invocation of dada.
- $err_out: The err matrix estimated from the output of dada. NULL if err_function not provided.
- $opts: A list of the dada_opts used for this invocation of dada.
- $call: The function call used for this invocation of dada.

See Also

dada

---

derep-class *A class representing dereplicated sequences*

Description

A list with the following three members.

- $uniques: Named integer vector. Named by the unique sequence, valued by abundance.
- $quals: Numeric matrix of average quality scores by position for each unique. Uniques are rows, positions are cols.
- $map: Integer vector of length the number of reads, and value the index (in $uniques) of the unique to which that read was assigned.

This can be created from a FastQ sequence file using derepFastq

See Also

derepFastq

---

derepFastq *Read in and dereplicate a fastq file.*

Description

A custom interface to FastqStreamer for dereplicating amplicon sequences from fastq or compressed fastq files, while also controlling peak memory requirement to support large files.

Usage

```
derepFastq(fls, n = 1e+06, verbose = FALSE)
```

Arguments

fls	(Required). character. The file path(s) to the fastq or fastq.gz file(s). Actually, any file format supported by FastqStreamer.
n	(Optional). numeric(1). The maximum number of records (reads) to parse and dereplicate at any one time. This controls the peak memory requirement so that large fastq files are supported. Default is 1e6, one-million reads. See FastqStreamer for details on this parameter, which is passed on.
verbose	(Optional). Default FALSE. If TRUE, throw standard R messages on the intermittent and final status of the dereplication.

Value

A derep-class object or list of such objects.

Examples

```
# Test that chunk-size, `n`, does not affect the result.
testFastq = system.file("extdata", "sam1F.fastq.gz", package="dada2")
derep1 = derepFastq(testFastq, verbose = TRUE)
derep1.35 = derepFastq(testFastq, 35, TRUE)
all.equal(getUniques(derep1), getUniques(derep1.35)[names(getUniques(derep1))])
```

---

errBalancedF *An empirical error matrix.*

Description

A dataset containing the error matrix estimated by DADA2 from the forward reads of the Illumina Miseq 2x250 sequenced Balanced mock community (see manuscript).

Format

A numerical matrix with 16 rows and 41 columns. Rows correspond to the 16 transition (eg. A2A, A2C, ...) Columns correspond to consensus quality scores 0 to 40.

---

errBalancedR *An empirical error matrix.*

Description

A dataset containing the error matrix estimated by DADA2 from the reverse reads of the Illumina Miseq 2x250 sequenced Balanced mock community (see manuscript).

Format

A numerical matrix with 16 rows and 41 columns. Rows correspond to the 16 transition (eg. A2A, A2C, ...) Columns correspond to consensus quality scores 0 to 40.

errExtremeF *An empirical error matrix.*

Description

A dataset containing the error matrix estimated by DADA2 from the forward reads of the Illumina Miseq 2x250 sequenced Extreme mock community (see manuscript).

Format

A numerical matrix with 16 rows and 41 columns. Rows correspond to the 16 transition (eg. A2A, A2C, ...) Columns correspond to consensus quality scores 0 to 40.

errExtremeR *An empirical error matrix.*

Description

A dataset containing the error matrix estimated by DADA2 from the reverse reads of the Illumina Miseq 2x250 sequenced Extreme mock community (see manuscript).

Format

A numerical matrix with 16 rows and 41 columns. Rows correspond to the 16 transition (eg. A2A, A2C, ...) Columns correspond to consensus quality scores 0 to 40.

errHmpF *An empirical error matrix.*

Description

A dataset containing the error matrix estimated by DADA2 from the forward reads of the Illumina Miseq 2x250 sequenced HMP mock community (see manuscript).

Format

A numerical matrix with 16 rows and 41 columns. Rows correspond to the 16 transition (eg. A2A, A2C, ...) Columns correspond to consensus quality scores 0 to 40.

errHmpR *An empirical error matrix.*

Description

A dataset containing the error matrix estimated by DADA2 from the reverse reads of the Illumina Miseq 2x250 sequenced HMP mock community (see manuscript).

Format

A numerical matrix with 16 rows and 41 columns. Rows correspond to the 16 transition (eg. A2A, A2C, ...) Columns correspond to consensus quality scores 0 to 40.

fastqFilter *Filter and trim a fastq file.*

Description

fastqFilter takes an input fastq file (can be compressed), filters it based on several user-definable criteria, and outputs those reads which pass the filter to a new fastq file (also can be compressed). Several functions in the ShortRead package are leveraged to do this filtering.

Usage

```
fastqFilter(fn, fout, truncQ = 2, truncLen = 0, maxLen = Inf,
  minLen = 20, trimLeft = 0, maxN = 0, minQ = 0, maxEE = Inf,
  rm.phix = TRUE, primer.fwd = NULL, n = 1e+06, OMP = TRUE,
  compress = TRUE, verbose = FALSE, ...)
```

Arguments

fn	(Required). The path to the input fastq file.
fout	(Required). The path to the output file. Note that by default (compress=TRUE) the output fastq file is gzipped.
truncQ	(Optional). Default 2. Truncate reads at the first instance of a quality score less than or equal to truncQ.
truncLen	(Optional). Default 0 (no truncation). Truncate reads after truncLen bases. Reads shorter than this are discarded.
maxLen	(Optional). Default Inf (no maximum). Remove reads with length greater than maxLen. maxLen is enforced on the raw reads.
minLen	(Optional). Default 20. Remove reads with length less than minLen. minLen is enforced after all other trimming and truncation.
trimLeft	(Optional). Default 0. The number of nucleotides to remove from the start of each read. If both truncLen and trimLeft are provided, filtered reads will have length truncLen-trimLeft.
maxN	(Optional). Default 0. After truncation, sequences with more than maxN Ns will be discarded. Note that dada currently does not allow Ns.

minQ	(Optional). Default 0. After truncation, reads contain a quality score below minQ will be discarded.
maxEE	(Optional). Default Inf (no EE filtering). After truncation, reads with higher than maxEE "expected errors" will be discarded. Expected errors are calculated from the nominal definition of the quality score: EE = sum(10^(-Q/10))
rm.phix	(Optional). Default TRUE. If TRUE, discard reads that match against the phiX genome, as determined by isPhiX.
primer.fwd	(Optional). Default NULL. A character string defining the forward primer. Only allows unambiguous nucleotides. The primer will be compared to the first len(primer.fwd) nucleotides at the start of the read. If there is not an exact match, the read is filtered out.
n	(Optional). The number of records (reads) to read in and filter at any one time. This controls the peak memory requirement so that very large fastq files are supported. Default is 1e6, one-million reads. See FastqStreamer for details.
OMP	(Optional). Default TRUE. Whether or not to use OMP multithreading when calling FastqStreamer. Set this to FALSE if calling this function within a parallelized chunk of code (eg. within mclapply).
compress	(Optional). Default TRUE. Whether the output fastq file should be gzip compressed.
verbose	(Optional). Default FALSE. Whether to output status messages.
...	(Optional). Arguments passed on to isPhiX.

Value

integer(2). The number of reads read in, and the number of reads that passed the filter and were output.

See Also

fastqPairedFilter FastqStreamer trimTails

Examples

```
testFastq = system.file("extdata", "sam1F.fastq.gz", package="dada2")
filtFastq <- tempfile(fileext=".fastq.gz")
fastqFilter(testFastq, filtFastq, maxN=0, maxEE=2)
fastqFilter(testFastq, filtFastq, trimLeft=10, truncLen=200, maxEE=2, verbose=TRUE)
```

---

fastqPairedFilter *Filters and trims paired forward and reverse fastq files.*

Description

fastqPairedFilter filters pairs of input fastq files (can be compressed) based on several user-definable criteria, and outputs those read pairs which pass the filter in **both** directions to two new fastq file (also can be compressed). Several functions in the ShortRead package are leveraged to do this filtering. The filtered forward/reverse reads remain identically ordered.

Usage

```
fastqPairedFilter(fn, fout, maxN = c(0, 0), truncQ = c(2, 2),
  truncLen = c(0, 0), maxLen = c(Inf, Inf), minLen = c(20, 20),
  trimLeft = c(0, 0), minQ = c(0, 0), maxEE = c(Inf, Inf),
  rm.phix = c(TRUE, TRUE), matchIDs = FALSE, primer.fwd = NULL,
  id.sep = "\\s", id.field = NULL, n = 1e+06, OMP = TRUE,
  compress = TRUE, verbose = FALSE, ...)
```

Arguments

fn	(Required). A character(2) naming the paths to the (forward,reverse) fastq files.
fout	(Required). A character(2) naming the paths to the (forward,reverse) output files. Note that by default (compress=TRUE) the output fastq files are gzipped. **FILTERING AND TRIMMING ARGUMENTS** If a length 1 vector is provided, the same parameter value is used for the forward and reverse reads. If a length 2 vector is provided, the first value is used for the forward reads, and the second for the reverse reads.
maxN	(Optional). Default 0. After truncation, sequences with more than maxN Ns will be discarded. Note that dada currently does not allow Ns.
truncQ	(Optional). Default 2. Truncate reads at the first instance of a quality score less than or equal to truncQ.
truncLen	(Optional). Default 0 (no truncation). Truncate reads after truncLen bases. Reads shorter than this are discarded.
maxLen	(Optional). Default Inf (no maximum). Remove reads with length greater than maxLen. maxLen is enforced on the raw reads.
minLen	(Optional). Default 20. Remove reads with length less than minLen. minLen is enforced after all other trimming and truncation.
trimLeft	(Optional). Default 0. The number of nucleotides to remove from the start of each read. If both truncLen and trimLeft are provided, filtered reads will have length truncLen-trimLeft.
minQ	(Optional). Default 0. After truncation, reads contain a quality score below minQ will be discarded.
maxEE	(Optional). Default Inf (no EE filtering). After truncation, reads with higher than maxEE "expected errors" will be discarded. Expected errors are calculated from the nominal definition of the quality score: EE = sum(10^(-Q/10))
rm.phix	(Optional). Default TRUE. If TRUE, discard reads that match against the phiX genome, as determined by isPhiX.
matchIDs	(Optional). Default FALSE. Whether to enforce matching between the id-line sequence identifiers of the forward and reverse fastq files. If TRUE, only paired reads that share id fields (see below) are output. If FALSE, no read ID checking is done. Note: matchIDs=FALSE essentially assumes matching order between forward and reverse reads. If that matched order is not present future processing steps may break (in particular mergePairs).
primer.fwd	(Optional). Default NULL. A character string defining the forward primer. Only allows unambiguous nucleotides. The primer will be compared to the first len(primer.fwd) nucleotides at the start of the forward and reverse reads. If there is not an exact match, the paired read is filtered out. If detected on the reverse read, the fwd/rev reads are swapped.

ID MATCHING ARGUMENTS

The following optional arguments enforce matching between the sequence identification strings in the forward and reverse reads, and can automatically detect and match ID fields in Illumina format, e.g: EAS139:136:FC706VJ:2:2104:15343:197393. ID matching is not required when using standard Illumina output fastq files.

`id.sep` (Optional). Default "\s" (white-space). The separator between fields in the id-line of the input fastq files. Passed to the `strsplit`.

`id.field` (Optional). Default NULL (automatic detection). The field of the id-line containing the sequence identifier. If NULL (the default) and matchIDs is TRUE, the function attempts to automatically detect the sequence identifier field under the assumption of Illumina formatted output.

`n` (Optional). The number of records (reads) to read in and filter at any one time. This controls the peak memory requirement so that very large fastq files are supported. Default is `1e6`, one-million reads. See `FastqStreamer` for details.

`OMP` (Optional). Default TRUE. Whether or not to use OMP multithreading when calling `FastqStreamer`. Set this to FALSE if calling this function within a parallelized chunk of code (eg. within `mclapply`).

`compress` (Optional). Default TRUE. Whether the output fastq files should be gzip compressed.

`verbose` (Optional). Default FALSE. Whether to output status messages.

`...` (Optional). Arguments passed on to `isPhiX`.

Value

`integer(2)`. The number of reads read in, and the number of reads that passed the filter and were output.

See Also

`fastqFilter` `FastqStreamer` `trimTails`

Examples

```
testFastqF = system.file("extdata", "sam1F.fastq.gz", package="dada2")
testFastqR = system.file("extdata", "sam1R.fastq.gz", package="dada2")
filtFastqF <- tempfile(fileext=".fastq.gz")
filtFastqR <- tempfile(fileext=".fastq.gz")
fastqPairedFilter(c(testFastqF, testFastqR), c(filtFastqF, filtFastqR), maxN=0, maxEE=2)
fastqPairedFilter(c(testFastqF, testFastqR), c(filtFastqF, filtFastqR), trimLeft=c(10, 20),
                  truncLen=c(240, 200), maxEE=2, rm.phix=TRUE, verbose=TRUE)
```

`filterAndTrim` *Filter and trim fastq file(s).*

Description

Filters and trims an input fastq file(s) (can be compressed) based on several user-definable criteria, and outputs fastq file(s) (compressed by default) containing those trimmed reads which passed the filters. Corresponding forward and reverse fastq file(s) can be provided as input, in which case filtering is performed on the forward and reverse reads independently, and both reads must pass for the read pair to be output.

Usage

```
filterAndTrim(fwd, filt, rev = NULL, filt.rev = NULL, compress = TRUE,
  truncQ = 2, truncLen = 0, trimLeft = 0, maxLen = Inf, minLen = 20,
  maxN = 0, minQ = 0, maxEE = Inf, rm.phix = TRUE, primer.fwd = NULL,
  matchIDs = FALSE, id.sep = "\\s", id.field = NULL,
  multithread = FALSE, n = 1e+05, OMP = TRUE, verbose = FALSE)
```

Arguments

`fwd`	(Required). `character`. The path(s) to the input fastq file(s).
`filt`	(Required). `character`. The path(s) to the output filtered file(s) corresponding to the `fwd` input files. If containing directory does not exist, it will be created.
`rev`	(Optional). Default NULL. The path(s) to the input reverse fastq file(s) from paired-end sequence data corresponding to those provided to the `fwd` argument. If NULL, the `fwd` files are processed as single-reads.
`filt.rev`	(Optional). Default NULL, but required if `rev` is provided. The path(s) to the output fastq file(s) corresponding to the `rev` input. Can also provide a directory, which if not existing will be created (how to differentiate between dir/file in len1 case?).
`compress`	(Optional). Default TRUE. If TRUE, the output fastq file(s) are gzipped.
	FILTERING AND TRIMMING PARAMETERS ———
	Note: When filtering paired reads... If a length 1 vector is provided, the same parameter value is used for the forward and reverse reads. If a length 2 vector is provided, the first value is used for the forward reads, and the second for the reverse reads.
`truncQ`	(Optional). Default 2. Truncate reads at the first instance of a quality score less than or equal to `truncQ`.
`truncLen`	(Optional). Default 0 (no truncation). Truncate reads after `truncLen` bases. Reads shorter than this are discarded.
`trimLeft`	(Optional). Default 0. The number of nucleotides to remove from the start of each read. If both `truncLen` and `trimLeft` are provided, filtered reads will have length `truncLen-trimLeft`.
`maxLen`	(Optional). Default Inf (no maximum). Remove reads with length greater than maxLen. maxLen is enforced **before** trimming and truncation.
`minLen`	(Optional). Default 20. Remove reads with length less than minLen. minLen is enforced **after** trimming and truncation.
`maxN`	(Optional). Default 0. After truncation, sequences with more than `maxN` Ns will be discarded. Note that dada does not allow Ns.
`minQ`	(Optional). Default 0. After truncation, reads contain a quality score less than `minQ` will be discarded.

maxEE (Optional). Default Inf (no EE filtering). After truncation, reads with higher than maxEE "expected errors" will be discarded. Expected errors are calculated from the nominal definition of the quality score: EE = sum(10^(-Q/10))

rm.phix (Optional). Default TRUE. If TRUE, discard reads that match against the phiX genome, as determined by isPhiX.

primer.fwd (Optional). Default NULL. Paired-read filtering only. A character string defining the forward primer. Only allows unambiguous nucleotides. The primer will be compared to the first len(primer.fwd) nucleotides at the start of the read. If there is not an exact match, the read is filtered out. For paired reads, the reverse read is also interrogated, and if the primer is detected on the reverse read, the forward/reverse reads are swapped.

matchIDs (Optional). Default FALSE. Paired-read filtering only. Whether to enforce matching between the id-line sequence identifiers of the forward and reverse fastq files. If TRUE, only paired reads that share id fields (see below) are output. If FALSE, no read ID checking is done. Note: matchIDs=FALSE essentially assumes matching order between forward and reverse reads. If that matched order is not present future processing steps may break (in particular mergePairs).

id.sep (Optional). Default "\s" (white-space). Paired-read filtering only. The separator between fields in the id-line of the input fastq files. Passed to the strsplit.

id.field (Optional). Default NULL (automatic detection). Paired-read filtering only. The field of the id-line containing the sequence identifier. If NULL (the default) and matchIDs is TRUE, the function attempts to automatically detect the sequence identifier field under the assumption of Illumina formatted output.

multithread (Optional). Default is FALSE. If TRUE, input files are filtered in parallel via mclapply. If an integer is provided, it is passed to the mc.cores argument of mclapply. Note that the parallelization here is by forking, and each process is loading another fastq file into memory. This option is ignored in Windows, as Windows does not support forking, with mc.cores set to 1. If memory is an issue, execute in a clean environment and reduce the chunk size n and/or the number of threads.

n (Optional). Default 1e5. The number of records (reads) to read in and filter at any one time. This controls the peak memory requirement so that very large fastq files are supported. See FastqStreamer for details.

OMP (Optional). Default TRUE. Whether or not to use OMP multithreading when calling FastqStreamer. Should be set to FALSE if calling this function within a parallelized chunk of code. If multithread=TRUE, this argument will be coerced to FALSE.

verbose (Optional). Default FALSE. Whether to output status messages.

Details

filterAndTrim is a multithreaded convenience interface for the fastqFilter and fastqPairedFilter filtering functions. Note that error messages and tracking are not handled gracefully when using the multithreading functionality. If errors arise, it is recommended to re-run without multithreading to troubleshoot the issue.

Value

Integer matrix. Returned invisibly (i.e. only if assigned to something). Rows correspond to the input files, columns record the reads.in and reads.out after filtering.

See Also

fastqFilter fastqPairedFilter FastqStreamer

Examples

```
testFastqs = c(system.file("extdata", "sam1F.fastq.gz", package="dada2"),
               system.file("extdata", "sam2F.fastq.gz", package="dada2"))
filtFastqs <- c(tempfile(fileext=".fastq.gz"), tempfile(fileext=".fastq.gz"))
filterAndTrim(testFastqs, filtFastqs, maxN=0, maxEE=2, verbose=TRUE)
filterAndTrim(testFastqs, filtFastqs, truncQ=2, truncLen=200, rm.phix=TRUE)
```

---

getDadaOpt *Get DADA options*

Description

Get DADA options

Usage

```
getDadaOpt(option = NULL)
```

Arguments

option	(Optional). Character. The DADA option(s) to get.

Value

Named list of option/value pairs. Returns NULL if an invalid option is requested.

See Also

setDadaOpt

Examples

```
getDadaOpt("BAND_SIZE")
getDadaOpt()
```

getErrors *Extract already computed error rates.*

Description

Extract already computed error rates.

Usage

```
getErrors(obj, detailed = FALSE, enforce = TRUE)
```

Arguments

obj	(Required). An R object with error rates. Supported objects: dada-class; list of dada-class; numeric matrix; named list with $err_out, $err_in, $trans.
detailed	(Optional). Default FALSE. If FALSE, an error rate matrix corresponding to $err_out is returned. If TRUE, a named list with $err_out, $err_in and $trans. $err_in and $trans can be NULL.
enforce	(Optional). Default TRUE. If TRUE, will check validity of $err_out and error if invalid or NULL.

Value

A numeric matrix of error rates. Or, if detailed=TRUE, a named list with $err_out, $err_in and $trans.

Examples

```
fl1 <- system.file("extdata", "sam1F.fastq.gz", package="dada2")
drp <- derepFastq(fl1)
dd <- dada(drp, err=NULL, selfConsist=TRUE)
err <- getErrors(dd)
```

getSequences *Get vector of sequences from input object.*

Description

This function extracts the sequences from several different data objects, including including dada-class and derep-class objects, as well as data.frame objects that have both $sequence and $abundance columns. This function wraps the getUniques function, but return only the names (i.e. the sequences). Can also be provided the file path to a fasta or fastq file, a taxonomy table, or a DNAStringSet object.

Usage

```
getSequences(object, collapse = FALSE, silence = TRUE)
```

Arguments

object	(Required). The object from which to extract the sequences.
collapse	(Optional). Default FALSE. Should duplicate sequences detected in object be collapsed together, thereby imposing uniqueness on non-unique input.
silence	(Optional). Default TRUE. Suppress reporting of the detection and merger of duplicated input sequences.

Value

character. A character vector of the sequences.

Examples

```
derep1 = derepFastq(system.file("extdata", "sam1F.fastq.gz", package="dada2"))
dada1 <- dada(derep1, err=tperr1)
getSequences(derep1)
getSequences(dada1)
getSequences(dada1$clustering)
```

---

getUniques *Get the uniques-vector from the input object.*

Description

This function extracts the uniques-vector from several different data objects, including dada-class and derep-class objects, as well as data.frame objects that have both $sequence and $abundance columns. The return value is an integer vector named by sequence and valued by abundance. If the input is already in uniques-vector format, that same vector will be returned.

Usage

```
getUniques(object, collapse = TRUE, silence = FALSE)
```

Arguments

object	(Required). The object from which to extract the uniques-vector.
collapse	(Optional). Default TRUE. Should duplicate sequences detected in object be collapsed together, thereby imposing uniqueness on non-unique input.
silence	(Optional). Default FALSE. Suppress reporting of the detection and merger of duplicated input sequences.

Value

integer. An integer vector named by unique sequence and valued by abundance.

Examples

```
derep1 = derepFastq(system.file("extdata", "sam1F.fastq.gz", package="dada2"))
dada1 <- dada(derep1, err=tperr1)
getUniques(derep1)
getUniques(dada1)
getUniques(dada1$clustering)
```

`inflateErr` *Inflates an error rate matrix by a specified factor, while accounting for saturation.*

Description

Error rates are "inflated" by the specified factor, while appropriately saturating so that rates cannot exceed 1. The formula is: new_err_rate <- err_rate * inflate / (1 + (inflate-1) * err_rate)

Usage

```
inflateErr(err, inflation, inflateSelfTransitions = FALSE)
```

Arguments

`err`	(Required). A numeric matrix of transition rates (16 rows, named "A2A", "A2C", ...).
`inflation`	(Required). The fold-factor by which to inflate the transition rates.
`inflateSelfTransitions`	(Optional). Default FALSE. If True, self-transitions (eg. A->A) are also inflated.

Value

An error rate matrix of the same dimensions as the input error rate matrix.

Examples

```
tperr2 <- inflateErr(tperr1, 2)
tperr3.all <- inflateErr(tperr1, 3, inflateSelfTransitions=TRUE)
```

`isBimera` *Determine if input sequence is a bimera of putative parent sequences.*

Description

This function attempts to find an exact bimera of the parent sequences that matches the input sequence. A bimera is a two-parent chimera, in which the left side is made up of one parent sequence, and the right-side made up of a second parent sequence. If an exact bimera is found TRUE is returned, otherwise FALSE. Bimeras that are one-off from exact are also identified if the allowOneOff argument is TRUE.

Usage

```
isBimera(sq, parents, allowOneOff = FALSE, minOneOffParentDistance = 4,
  maxShift = 16)
```

Arguments

sq	(Required). A character(1). The sequence being evaluated as a possible bimera.
parents	(Required). Character vector. A vector of possible "parent" sequence that could form the left and right sides of the bimera.
allowOneOff	(Optional). A logical(1). Default is FALSE. If FALSE, sq will be identified as a bimera if it is one mismatch or indel away from an exact bimera.
minOneOffParentDistance	(Optional). A numeric(1). Default is 4. Only sequences with at least this many mismatches to sq are considered as possible "parents" when flagging one-off bimeras. There is no such screen when identifying exact bimeras.
maxShift	(Optional). A numeric(1). Default is 16. Maximum shift allowed when aligning sequences to potential "parents".

Value

logical(1). TRUE if sq is a bimera of two of the parents. Otherwise FALSE.

See Also

isBimeraDenovo, removeBimeraDenovo

Examples

```
derep1 = derepFastq(system.file("extdata", "sam1F.fastq.gz", package="dada2"))
sqs1 <- getSequences(derep1)
isBimera(sqs1[[20]], sqs1[1:10])
```

---

isBimeraDenovo *Identify bimeras from collections of unique sequences.*

Description

This function is a wrapper around isBimera for collections of unique sequences (i.e. sequences with associated abundances). Each sequence is evaluated against a set of "parents" drawn from the sequence collection that are sufficiently more abundant than the sequence being evaluated. A logical vector is returned, with an entry for each input sequence indicating whether it was (was not) consistent with being a bimera of those more abundant "parents".

Usage

```
isBimeraDenovo(unqs, minFoldParentOverAbundance = 2, minParentAbundance = 8,
  allowOneOff = FALSE, minOneOffParentDistance = 4, maxShift = 16,
  multithread = FALSE, verbose = FALSE)
```

Arguments

unqs	(Required). A uniques-vector or any object that can be coerced into one with getUniques.
minFoldParentOverAbundance	(Optional). A numeric(1). Default is 2. Only sequences greater than this-fold more abundant than a sequence can be its "parents".
minParentAbundance	(Optional). A numeric(1). Default is 8. Only sequences at least this abundant can be "parents".
allowOneOff	(Optional). A logical(1). Default is FALSE. If FALSE, sequences that have one mismatch or indel to an exact bimera are also flagged as bimeric.
minOneOffParentDistance	(Optional). A numeric(1). Default is 4. Only sequences with at least this many mismatches to the potential bimeric sequence considered as possible "parents" when flagging one-off bimeras. There is no such screen when considering exact bimeras.
maxShift	(Optional). A numeric(1). Default is 16. Maximum shift allowed when aligning sequences to potential "parents".
multithread	(Optional). Default is FALSE. If TRUE, multithreading is enabled and the number of available threads is automatically determined. If an integer is provided, the number of threads to use is set by passing the argument on to mclapply.
verbose	(Optional). logical(1) indicating verbose text output. Default FALSE.

Value

logical of length the number of input unique sequences. TRUE if sequence is a bimera of more abundant "parent" sequences. Otherwise FALSE.

See Also

isBimera, removeBimeraDenovo

Examples

```
derep1 = derepFastq(system.file("extdata", "sam1F.fastq.gz", package="dada2"))
dada1 <- dada(derep1, err=tperr1, errorEstimationFunction=loessErrfun, selfConsist=TRUE)
isBimeraDenovo(dada1)
isBimeraDenovo(dada1$denoised, minFoldParentOverAbundance = 2, allowOneOff=TRUE)
```

---

isBimeraDenovoTable *Identify bimeras in a sequence table.*

Description

This function implements a table-specific version of de novo bimera detection. In short, bimeric sequences are flagged on a sample-by-sample basis. Then, a vote is performed for each sequence across all samples in which it appeared. If the sequence is flagged in a sufficiently high fraction of samples, it is identified as a bimera. A logical vector is returned, with an entry for each sequence in the table indicating whether it was identified as bimeric by this consensus procedure.

Usage

```
isBimeraDenovoTable(seqtab, minSampleFraction = 0.9, ignoreNNegatives = 1,
  minFoldParentOverAbundance = 1.5, minParentAbundance = 2,
  allowOneOff = FALSE, minOneOffParentDistance = 4, maxShift = 16,
  multithread = FALSE, verbose = FALSE)
```

Arguments

seqtab	(Required). A sequence table. That is, an integer matrix with colnames corresponding to DNA sequences.
minSampleFraction	(Optional). Default is 0.9. The fraction of samples in which a sequence must be flagged as bimeric in order for it to be classified as a bimera.
ignoreNNegatives	(Optional). Default is 1. The number of unflagged samples to ignore when evaluating whether the fraction of samples in which a sequence was flagged as a bimera exceeds minSampleFraction. The purpose of this parameter is to lower the threshold at which sequences found in few samples are flagged as bimeras.
minFoldParentOverAbundance	(Optional). Default is 1.5. Only sequences greater than this-fold more abundant than a sequence can be its "parents". Evaluated on a per-sample basis.
minParentAbundance	(Optional). Default is 2. Only sequences at least this abundant can be "parents". Evaluated on a per-sample basis.
allowOneOff	(Optional). Default is FALSE. If FALSE, sequences that have one mismatch or indel to an exact bimera are also flagged as bimeric.
minOneOffParentDistance	(Optional). Default is 4. Only sequences with at least this many mismatches to the potential bimeric sequence considered as possible "parents" when flagging one-off bimeras. There is no such screen when considering exact bimeras.
maxShift	(Optional). Default is 16. Maximum shift allowed when aligning sequences to potential "parents".
multithread	(Optional). Default is FALSE. If TRUE, multithreading is enabled. NOT YET IMPLEMENTED.
verbose	(Optional). Default FALSE. Print verbose text output.

Value

logical of length equal to the number of sequences in the input table. TRUE if sequence is identified as a bimera. Otherwise FALSE.

See Also

isBimera, removeBimeraDenovo

Examples

```
derep1 = derepFastq(system.file("extdata", "sam1F.fastq.gz", package="dada2"))
derep2 = derepFastq(system.file("extdata", "sam2F.fastq.gz", package="dada2"))
dd <- dada(list(derep1,derep2), err=NULL, errorEstimationFunction=loessErrfun, selfConsist=TRUE)
seqtab <- makeSequenceTable(dd)
```

```
isBimeraDenovoTable(seqtab)
isBimeraDenovoTable(seqtab, allowOneOff=TRUE, minSampleFraction=0.5)
```

`isPhiX` *Determine if input sequence(s) match the phiX genome.*

Description

This function compares the word-profile of the input sequences to the phiX genome, and the reverse complement of the phiX genome. If enough exactly matching words are found, the sequence is flagged.

Usage

```
isPhiX(seqs, wordSize = 16, minMatches = 2, nonOverlapping = TRUE)
```

Arguments

`seqs`	(Required). A `character` vector of A/C/G/T sequences.
`wordSize`	(Optional). Default 16. The size of the words to use for comparison.
`minMatches`	(Optional). Default 2. The minimum number of words in the input sequences that must match the phiX genome (or its reverse complement) for the sequence to be flagged.
`nonOverlapping`	(Optional). Default TRUE. If TRUE, only non-overlapping matching words are counted.

Value

`logical(1)`. TRUE if sequence matched the phiX genome.

See Also

`fastqFilter`, `fastqPairedFilter`

Examples

```
derep1 = derepFastq(system.file("extdata", "sam1F.fastq.gz", package="dada2"))
sqs1 <- getSequences(derep1)
isPhiX(sqs1)
isPhiX(sqs1, wordSize=20,  minMatches=1)
```

`isShiftDenovo` *Identify sequences that are identical to a more abundant sequence up to an overall shift.*

Description

This function is a wrapper around isShift for collections of unique sequences. Each unique sequence is evaluated against a set of "parents" drawn from the sequence collection that are more abundant than the sequence being evaluated.

Usage

```
isShiftDenovo(unqs, minOverlap = 20, flagSubseqs = FALSE, verbose = FALSE)
```

Arguments

`unqs`	(Required). A `uniques-vector` or any object that can be coerced into one with `getUniques`.
`minOverlap`	(Optional). A `numeric(1)`. Default is 20. Minimum overlap required to call something a shift.
`flagSubseqs`	(Optional). A `logical(1)`. Default is FALSE. Whether or not to flag strict subsequences as shifts.
`verbose`	(Optional). `logical(1)` indicating verbose text output. Default FALSE.

Value

`logical` of length the number of input unique sequences. TRUE if sequence is an exact shift of a more abundant sequence. Otherwise FALSE.

See Also

`isBimera`

Examples

```
derep1 = derepFastq(system.file("extdata", "sam1F.fastq.gz", package="dada2"))
dada1 <- dada(derep1, err=tperr1, errorEstimationFunction=loessErrfun, selfConsist=TRUE)
isShiftDenovo(dada1)
isShiftDenovo(dada1$denoised, minOverlap=50, verbose=TRUE)
```

`learnErrors` *Learns the error rates from an input list, or vector, of file names or a list of* `derep-class` *objects.*

Description

Error rates are learned by alternating between sample inference and error rate estimation until convergence. Sample inferences is performed by the `dada` function. Error rate estimation is performed by `errorEstimationFunction`. The output of this function serves as input to the dada function call as the `err` parameter.

Usage

```
learnErrors(fls, nbases = 1e+08, nreads = NULL,
  errorEstimationFunction = loessErrfun, multithread = FALSE,
  randomize = FALSE, MAX_CONSIST = 10, OMEGA_C = 0, verbose = FALSE,
  ...)
```

Arguments

`fls`	(Required). `character`. The file path(s) to the fastq, fastq.gz file(s), or any file format supported by `FastqStreamer`. A list of derep-class ojects can also be provided.
`nbases`	(Optional). Default 1e8. The minimum number of total bases to use for error rate learning. Samples are read into memory until at least this number of total bases has been reached, or all provided samples have been read in.
`nreads`	(Optional). Default NULL. DEPRECATED. Please update your code to use the nbases parameter.
`errorEstimationFunction`	(Optional). Function. Default `loessErrfun`. `errorEstimationFunction` is computed on the matrix of observed transitions after each sample inference step in order to generate the new matrix of estimated error rates.
`multithread`	(Optional). Default is FALSE. If TRUE, multithreading is enabled and the number of available threads is automatically determined. If an integer is provided, the number of threads to use is set by passing the argument on to `setThreadOptions`.
`randomize`	(Optional). Default FALSE. If FALSE, samples are read in the provided order until enough reads are obtained. If TRUE, samples are picked at random from those provided.
`MAX_CONSIST`	(Optional). Default 10. The maximum number of times to step through the self-consistency loop. If convergence was not reached in MAX_CONSIST steps, the estimated error rates in the last step are returned.
`OMEGA_C`	(Optional). Default 0. The threshold at which unique sequences inferred to contain errors are corrected in the final output, and used to estimate the error rates (see more at `setDadaOpt`). For reasons of convergence, and because it is more conservative, it is recommended to set this value to 0, which means that all reads are counted and contribute to estimating the error rates.
`verbose`	(Optional). Default FALSE. Print verbose text output. More fine-grained control is available by providing an integer argument.

- 0: Silence. No text output
- 1: Minimal text output (same as FALSE).
- 2: Detailed output (same as TRUE).

... (Optional). Additional arguments will be passed on to the dada function.

Value

A named list with three entries: $err_out: A numeric matrix with the learned error rates. $err_in: The initialization error rates (unimportant). $trans: A feature table of observed transitions for each type (eg. A->C) and quality score.

See Also

derepFastq, plotErrors, loessErrfun, dada

Examples

```
fl1 <- system.file("extdata", "sam1F.fastq.gz", package="dada2")
fl2 <- system.file("extdata", "sam2F.fastq.gz", package="dada2")
err <- learnErrors(c(fl1, fl2))
err <- learnErrors(c(fl1, fl2), nreads=50000, randomize=TRUE)
# Using a list of derep-class objects
dereps <- derepFastq(c(fl1, fl2))
err <- learnErrors(dereps, multithread=TRUE, randomize=TRUE, MAX_CONSIST=20)
```

---

loessErrfun *Use a loess fit to estimate error rates from transition counts.*

Description

This function accepts a matrix of observed transitions, with each transition corresponding to a row (eg. row 2 = A->C) and each column to a quality score (eg. col 31 = Q30). It returns a matrix of estimated error rates of the same shape. Error rates are estimates by a loess fit of the observed rates of each transition as a function of the quality score. Self-transitions (i.e. A->A) are taken to be the left-over probability.

Usage

```
loessErrfun(trans)
```

Arguments

trans (Required). A matrix of the observed transition counts. Must be 16 rows, with the rows named "A2A", "A2C", ...

Value

A numeric matrix with 16 rows and the same number of columns as trans. The estimated error rates for each transition (row, eg. "A2C") and quality score (column, eg. 31), as determined by loess smoothing over the quality scores within each transition category.

Examples

```
derep1 <- derepFastq(system.file("extdata", "sam1F.fastq.gz", package="dada2"))
dada1 <- dada(derep1, err=tperr1)
err.new <- loessErrfun(dada1$trans)
```

`makeSequenceTable` *Construct a sample-by-sequence observation matrix.*

Description

This function constructs a sequence table (analogous to an OTU table) from the provided list of samples.

Usage

```
makeSequenceTable(samples, orderBy = "abundance")
```

Arguments

`samples`	(Required). A `list` of the samples to include in the sequence table. Samples can be provided in any format that can be processed by `getUniques`. Sample names are propagated to the rownames of the sequence table.
`orderBy`	(Optional). `character(1)`. Default "abundance". Specifies how the sequences (columns) of the returned table should be ordered (decreasing). Valid values: "abundance", "nsamples", NULL.

Value

Named integer matrix. A row for each sample, and a column for each unique sequence across all the samples. Note that the columns are named by the sequence which can make display a little unwieldy.

See Also

`dada`, `getUniques`

Examples

```
derep1 <- derepFastq(system.file("extdata", "sam1F.fastq.gz", package="dada2"))
derep2 <- derepFastq(system.file("extdata", "sam2F.fastq.gz", package="dada2"))
dada1 <- dada(derep1, tperr1)
dada2 <- dada(derep2, tperr1)
makeSequenceTable(list(sample1=dada1, sample2=dada2))
```

mergePairs *Merge denoised forward and reverse reads.*

Description

This function attempts to merge each denoised pair of forward and reverse reads, rejecting any pairs which do not sufficiently overlap or which contain too many (>0 by default) mismatches in the overlap region. Note: This function assumes that the fastq files for the forward and reverse reads were in the same order.

Usage

```
mergePairs(dadaF, derepF, dadaR, derepR, minOverlap = 12, maxMismatch = 0,
  returnRejects = FALSE, propagateCol = character(0),
  justConcatenate = FALSE, trimOverhang = FALSE, verbose = FALSE, ...)
```

Arguments

dadaF	(Required). A dada-class object, or a list of such objects. The dada-class object(s) generated by denoising the forward reads.
derepF	(Required). A derep-class object, or a list of such objects. The derep-class object(s) used as input to the the dada function when denoising the forward reads.
dadaR	(Required). A dada-class object, or a list of such objects. The dada-class object(s) generated by denoising the reverse reads.
derepR	(Required). A derep-class object, or a list of such objects. The derep-class object(s) used as input to the the dada function when denoising the reverse reads.
minOverlap	(Optional). Default 12. The minimum length of the overlap required for merging the forward and reverse reads.
maxMismatch	(Optional). Default 0. The maximum mismatches allowed in the overlap region.
returnRejects	(Optional). Default FALSE. If TRUE, the pairs that that were rejected based on mismatches in the overlap region are retained in the return data.frame.
propagateCol	(Optional). character. Default character(0). The return data.frame will include values from columns in the $clustering data.frame of the provided dada-class objects with the provided names.
justConcatenate	(Optional). Default FALSE. If TRUE, the forward and reverse-complemented reverse read are concatenated rather than merged, with a NNNNNNNNNN (10 Ns) spacer inserted between them.
trimOverhang	(Optional). Default FALSE. If TRUE, "overhangs" in the alignment between the forwards and reverse read are trimmed off. "Overhangs" are when the reverse read extends past the start of the forward read, and vice-versa, as can happen when reads are longer than the amplicon and read into the other-direction primer region.
verbose	(Optional). Default FALSE. If TRUE, a summary of the function results are printed to standard output.
...	(Optional). Further arguments to pass on to nwalign. By default, mergePairs uses alignment parameters that hevaily penalizes mismatches and gaps when aligning the forward and reverse sequences.

Value

A `data.frame`, or a list of `data.frames`.

The return `data.frame`(s) has a row for each unique pairing of forward/reverse denoised sequences, and the following columns:

- `$abundance`: Number of reads corresponding to this forward/reverse combination.
- `$sequence`: The merged sequence.
- `$forward`: The index of the forward denoised sequence.
- `$reverse`: The index of the reverse denoised sequence.
- `$nmatch`: Number of matches nts in the overlap region.
- `$nmismatch`: Number of mismatches in the overlap region.
- `$nindel`: Number of indels in the overlap region.
- `$prefer`: The sequence used for the overlap region. 1=forward; 2=reverse.
- `$accept`: TRUE if overlap between forward and reverse denoised sequences was at least `minOverlap` and had at most `maxMismatch` differences. FALSE otherwise.
- `$...`: Additional columns specified in `propagateCol`.

A list of data.frames are returned if a list of input objects was provided.

See Also

`derepFastq`, `dada`, `fastqPairedFilter`

Examples

```
derepF = derepFastq(system.file("extdata", "sam1F.fastq.gz", package="dada2"))
derepR = derepFastq(system.file("extdata", "sam1R.fastq.gz", package="dada2"))
dadaF <- dada(derepF, err=tperr1, errorEstimationFunction=loessErrfun, selfConsist=TRUE)
dadaR <- dada(derepR, err=tperr1, errorEstimationFunction=loessErrfun, selfConsist=TRUE)
mergePairs(dadaF, derepF, dadaR, derepR)
mergePairs(dadaF, derepF, dadaR, derepR, returnRejects=TRUE, propagateCol=c("n0", "birth_ham"))
mergePairs(dadaF, derepF, dadaR, derepR, justConcatenate=TRUE)
```

---

mergePairsByID *Merge forward and reverse reads after DADA denoising, even if reads were not originally ordered together.*

Description

This function attempts to merge each pair of denoised forward and reverse reads, rejecting any which do not sufficiently overlap or which contain too many (>0 by default) mismatches in the overlap region. Note: This function does not assume that the fastq files for the forward and reverse reads were in the same order. If they are already in the same order, use `mergePairs`.

Usage

```
mergePairsByID(dadaF, derepF, srF, dadaR, derepR, srR, minOverlap = 20,
  maxMismatch = 0, returnRejects = FALSE, idRegExpr = c("\\s.+$", ""),
  includeCol = character(0), justConcatenate = FALSE, verbose = FALSE)
```

Arguments

dadaF	(Required). A dada-class object. The output of dada() function on the forward reads.
derepF	(Required). A derep-class object. The derep-class object returned by derepFastq() that was used as the input to the dada-class object passed to the dadaF argument.
srF	(Required). The trimmed and filtered forward reads that you used as input for derepFastq. More generally, this is an object that inherits from the ShortRead-class. In most cases this will be ShortReadQ-class. Objects from this class are the result of readFastq. Alternatively, this can be a character string that provides the path to your forward reads fastq file.
dadaR	(Required). A dada-class object. The output of dada() function on the reverse reads.
derepR	(Required). A derep-class object. See derepF description, but for the reverse reads.
srR	(Required). See srF description, but in this case provide for the reverse reads.
minOverlap	(Optional). A numeric(1) of the minimum length of the overlap (in nucleotides) required for merging the forward and reverse reads. Default is 20.
maxMismatch	(Optional). A numeric(1) of the maximum mismatches allowed in the overlap region. Default is 0 (i.e. only exact matches in the overlap region are accepted).
returnRejects	(Optional). A logical(1). Default is FALSE. If TRUE, the pairs that that were rejected based on mismatches in the overlap region are retained in the return data.frame.
idRegExpr	(Optional). A length 2 character() vector. This is passed along in order as the first two arguments to a gsub call that defines how each read id is parsed. The default is c("\s.+$", ""), which is a gsub directive to keep the id string from the beginning up to but not including the first space. For some sequencing platforms and/or read ID schemes, an alternative parsing of the IDs may be appropriate.
includeCol	(Optional). character. Default is character(0). The returned data.table will include columns with names specified by the dada-class$clustering data.frame.
justConcatenate	(Optional). NOT CURRENTLY SUPPORTED. logical(1), Default FALSE. If TRUE, the forward and reverse-complemented reverse read are concatenated rather than merged, with a NNNNNNNNNN (10 Ns) spacer inserted between them.
verbose	(Optional). logical(1) indicating verbose text output. Default FALSE.

Details

Not yet implemented: Use of the concatenate option will result in concatenating forward and reverse reads without attempting a merge/alignment step.

Value

A data.frame with a row for each unique pairing of forward/reverse denoised sequences, and the following columns:

- $abundance: Number of reads corresponding to this forward/reverse combination.

- $sequence: The merged sequence.
- $forward: The index of the forward denoised sequence.
- $reverse: The index of the reverse denoised sequence.
- $nmatch: Number of matches nts in the overlap region.
- $nmismatch: Number of mismatches in the overlap region.
- $nindel: Number of indels in the overlap region.
- $prefer: The sequence used for the overlap region. 1=forward; 2=reverse.
- $accept: TRUE if overlap between forward and reverse denoised sequences was at least minOverlap and had at most maxMismatch differences. FALSE otherwise.
- $...: Additional columns specified in propagateCol.

See Also

derepFastq, dada

Examples

```
# For the following example files, there are two ways to merge denoised directions.
# Because the read sequences are in order, `mergePairs()` works.
# `mergePairsByID` always works,
# because it uses the read IDs to match denoised pairs.
exFileF = system.file("extdata", "sam1F.fastq.gz", package="dada2")
exFileR = system.file("extdata", "sam1R.fastq.gz", package="dada2")
srF = ShortRead::readFastq(exFileF)
srR = ShortRead::readFastq(exFileR)
derepF = derepFastq(exFileF)
derepR = derepFastq(exFileR)
dadaF <- dada(derepF, err=tperr1, errorEstimationFunction=loessErrfun, selfConsist=TRUE)
dadaR <- dada(derepR, err=tperr1, errorEstimationFunction=loessErrfun, selfConsist=TRUE)
# Run and compare
ex1time = system.time({
ex1 <- mergePairs(dadaF, derepF, dadaR, derepR, verbose = TRUE)
    ex1 <- data.table::data.table(ex1)
 })
ex1time
# The new function, based on read IDs.
ex2time = system.time({
  ex2 = dada2:::mergePairsByID(dadaF = dadaF, derepF = derepF, srF = srF,
                       dadaR = dadaR, derepR = derepR, srR = srR, verbose = TRUE)
})
ex2time
# Compare results (should be identical)
ex2[(accept)]
data.table::setkey(ex2, sequence)
ex2[(accept), list(abundance = sum(abundance)), by = sequence]
# Same sequence set (exactly)
setequal(x = ex1$sequence,
         y = ex2[(accept)]$sequence)
# Test concatenation functionality
ex1cattime = system.time({
ex1cat <- mergePairs(dadaF, derepF, dadaR, derepR, justConcatenate = TRUE, verbose = TRUE)
sapply(ex1cat, class)
  # need to convert to a character
  ex1cat$sequence <- unlist(ex1cat$sequence)
```

```
  ex1cat <- data.table::data.table(ex1cat)
})
ex1cattime
ex2cattime = system.time({
  ex2cat <- dada2:::mergePairsByID(dadaF = dadaF, derepF = derepF, srF = srF,
                           dadaR = dadaR, derepR = derepR, srR = srR,
                           justConcatenate = TRUE, verbose = TRUE)
})
ex2cattime
ex2cat[(accept)]
# Compare results (should be identical)
data.table::setkey(ex1cat, sequence)
ex1cat[(accept), list(abundance = sum(abundance)), by = sequence]
data.table::setkey(ex2cat, sequence)
ex2cat[(accept), list(abundance = sum(abundance)), by = sequence]
# Same sequence set (exactly)
setequal(x = ex1cat$sequence,
         y = ex2cat$sequence)
intersect(x = ex1cat$sequence,
          y = ex2cat$sequence)
ex1cat[, nchar(sequence)]
ex2cat[, nchar(sequence)]
```

`mergeSequenceTables` *Merge two or more sample-by-sequence observation matrices.*

Description

This function combines sequence tables together into one merged sequences table.

Usage

```
mergeSequenceTables(table1 = NULL, table2 = NULL, ..., tables = NULL,
  repeats = "error", orderBy = "abundance")
```

Arguments

`table1`	(Optional, default=NULL). Named integer matrix. Rownames correspond to samples and column names correspond to sequences. The output of `makeSequenceTable`.
`table2`	(Optional, default=NULL). Named integer matrix. Rownames correspond to samples and column names correspond to sequences. The output of `makeSequenceTable`.
`...`	(Optional). Additional sequence tables.
`tables`	(Optional, default=NULL). Either a list of sequence tables, or a list/vector of RDS filenames corresponding to sequence tables. If provided, `table1`, `table2`, and any additional arguments will be ignored.
`repeats`	(Optional). Default "error". Specifies how merging should proceed in the presence of repeated sample names. Valid values: "error", "sum". If "sum", then samples with the same name are summed together in the merged table.
`orderBy`	(Optional). `character(1)`. Default "abundance". Specifies how the sequences (columns) of the returned table should be ordered (decreasing). Valid values: "abundance", "nsamples", NULL.

Value

Named integer matrix. A row for each sample, and a column for each unique sequence across all the samples. Note that the columns are named by the sequence which can make display unwieldy.

See Also

makeSequenceTable

Examples

```
## Not run:
 mergetab <- mergeSequenceTables(seqtab1, seqtab2, seqtab3) # unnamed arguments are assumed to be individual
  input_tables <- list(seqtab1, seqtab2, seqtab3)
  mergetab <- mergeSequenceTables(tables=input_tables) # list of sequence tables
  files <- c(file1, file2, file3)
  mergetab <- mergeSequenceTables(tables=files) # vector of filenames

## End(Not run)
```

---

names<-,dada,ANY-method

Deactivate renaming of dada-class objects.

Description

Deactivate renaming of dada-class objects.

Usage

```
## S4 replacement method for signature 'dada,ANY'
names(x) <- value
```

Arguments

x	an R object.
value	a character vector of up to the same length as x, or NULL.

Value

NULL.

names<-,derep,ANY-method

Deactivate renaming of derep-class objects.

Description

Deactivate renaming of derep-class objects.

Usage

```
## S4 replacement method for signature 'derep,ANY'
names(x) <- value
```

Arguments

x	an R object.
value	a character vector of up to the same length as x, or NULL.

Value

NULL.

noqualErrfun

Estimate error rates for each type of transition while ignoring quality scores.

Description

This function accepts a matrix of observed transitions, groups together all observed transitions regardless of quality scores, and estimates the error rate for that transition as the observed fraction of those transitions. This can be used in place of the default loessErrfun when calling learnErrors or link{dada} with the effect that quality scores will be effectively ignored.

Usage

```
noqualErrfun(trans, pseudocount = 1)
```

Arguments

trans	(Required). A matrix of the observed transition counts. Must be 16 rows, with the rows named "A2A", "A2C", ...
pseudocount	(Optional). Default 1. Added to each type of transition.

Value

A numeric matrix with 16 rows and the same number of columns as trans. The estimated error rates for each transition (row, eg. "A2C") are identical across all columns (which correspond to quality scores).

Examples

```
fl1 <- system.file("extdata", "sam1F.fastq.gz", package="dada2")
err.noqual <- learnErrors(fl1, errorEstimationFunction=noqualErrfun)
```

---

nwalign *Needleman-Wunsch alignment.*

Description

This function performs a Needleman-Wunsch alignment between two sequences.

Usage

```
nwalign(s1, s2, match = getDadaOpt("MATCH"),
  mismatch = getDadaOpt("MISMATCH"), gap = getDadaOpt("GAP_PENALTY"),
  homo_gap = NULL, band = -1, endsfree = TRUE, vec = FALSE)
```

Arguments

s1	(Required). character(1). The first sequence to align. A/C/G/T only.
s2	(Required). character(1). The second sequence to align. A/C/G/T only.
match	(Optional). numeric(1). Default is getDadaOpt("MATCH"). The score of a match in the alignment.
mismatch	(Optional). numeric(1). Default is getDadaOpt("MISMATCH"). The score of a mismatch in the alignment.
gap	(Optional). numeric(1). Default is getDadaOpt("GAP_PENALTY"). The alignment gap penalty. Should be negative.
homo_gap	(Optional). numeric(1). Default NULL (no special homopolymer penalty). The alignment gap penalty within homopolymer regions. Should be negative.
band	(Optional). numeric(1). Default -1 (no banding). The Needleman-Wunsch alignment can be banded. This value specifies the radius of that band. Set band = -1 to turn off banding.
endsfree	(Optional). logical(1). Default TRUE. Allow unpenalized gaps at the ends of the alignment.
vec	(Optional). logical(1). Default FALSE. Use DADA2's vectorized aligner instead of standard DP matrix. Not intended for long sequences (>1kb).

Value

character(2). The aligned sequences.

Examples

```
sq1 <- "CTAATACATGCAAGTCGAGCGAGTCTGCCTTGAAGATCGGAGTGCTTGCACTCTGTGAAACAAGATA"
sq2 <- "TTAACACATGCAAGTCGAACGGAAAGGCCAGTGCTTGCACTGGTACTCGAGTGGCGAACGGGTGAGT"
nwalign(sq1, sq2)
nwalign(sq1, sq2, band=16)
```

nwhamming *Hamming distance after Needlman-Wunsch alignment.*

Description

This function performs a Needleman-Wunsch alignment between two sequences, and then counts the number of mismatches and indels in that alignment. End gaps are not included in this count.

Usage

```
nwhamming(s1, s2, ...)
```

Arguments

s1	(Required). character(1). The first sequence to align. A/C/G/T only.
s2	(Required). character(1). The second sequence to align. A/C/G/T only.
...	(Optional). Further arguments to pass on to nwalign.

Value

integer(1). The total number of mismatches and gaps, excluding gaps at the beginning and end of the alignment.

Examples

```
 sq1 <- "CTAATACATGCAAGTCGAGCGAGTCTGCCTTGAAGATCGGAGTGCTTGCACTCTGTGAAACAAGATA"
 sq2 <- "TTAACACATGCAAGTCGAACGGAAAGGCCAGTGCTTGCACTGGTACTCGAGTGGCGAACGGGTGAGT"
nwhamming(sq1, sq2)
nwhamming(sq1, sq2, band=16)
```

plotErrors *Plot observed and estimated error rates.*

Description

This function plots the observed frequency of each transition (eg. A->C) as a function of the associated quality score. It also plots the final estimated error rates (if they exist). The initial input rates and the expected error rates under the nominal definition of quality scores can also be shown.

Usage

```
plotErrors(dq, nti = c("A", "C", "G", "T"), ntj = c("A", "C", "G", "T"),
  obs = TRUE, err_out = TRUE, err_in = FALSE, nominalQ = FALSE)
```

Arguments

dq	(Required). An object from which error rates can be extracted. Valid inputs are coercible by getErrors. This includes the output of the dada and learnErrors functions.
nti	(Optional). Default c("A","C","G","T"). Some combination of the 4 DNA nucleotides.
ntj	(Optional). Default c("A","C","G","T"). Some combination of the 4 DNA nucleotides. The error rates from nti->ntj will be plotted. If multiple nti or ntj are chosen, error rates from each-to-each will be plotted in a grid.
obs	(Optional). Default TRUE. If TRUE, the observed error rates are plotted as points.
err_out	(Optional). Default TRUE. If TRUE, plot the output error rates (solid line).
err_in	(Optional). Default FALSE. If TRUE, plot the input error rates (dashed line).
nominalQ	(Optional). Default FALSE. If TRUE, plot the expected error rates (red line) if quality scores exactly matched their nominal definition: Q = -10 log10(p_err).

Value

A ggplot2 object. Will be rendered to default device if printed, or can be stored and further modified. See ggsave for additional options.

See Also

learnErrors, getErrors

Examples

```
derep1 = derepFastq(system.file("extdata", "sam1F.fastq.gz", package="dada2"), verbose = TRUE)
dada1 <- dada(derep1, err = inflateErr(tperr1, 2), errorEstimationFunction = loessErrfun)
plotErrors(dada1)
plotErrors(dada1, "A", "C")
plotErrors(dada1, nti="A", ntj=c("A","C","G","T"), err_in=TRUE, nominalQ=TRUE)
```

---

plotQualityProfile *Plot quality profile of a fastq file.*

Description

This function plots a visual summary of the distribution of quality scores as a function of sequence position for the input fastq file(s).

Usage

```
plotQualityProfile(fl, n = 5e+05, aggregate = FALSE)
```

Arguments

fl	(Required). character. File path(s) to fastq or fastq.gz file(s).
n	(Optional). Default 500,000. The number of records to sample from the fastq file.
aggregate	(Optional). Default FALSE. If TRUE, compute an aggregate quality profile for all fastq files provided.

Details

The distribution of quality scores at each position is shown as a grey-scale heat map, with dark colors corresponding to higher frequency. The plotted lines show positional summary statistics: green is the mean, orange is the median, and the dashed orange lines are the 25th and 75th quantiles.

If the sequences vary in length, a red line will be plotted showing the percentage of reads that extend to at least that position.

Value

A ggplot2 object. Will be rendered to default device if printed, or can be stored and further modified. See ggsave for additional options.

Examples

```
plotQualityProfile(system.file("extdata", "sam1F.fastq.gz", package="dada2"))
```

---

removeBimeraDenovo *Remove bimeras from collections of unique sequences.*

Description

This function is a convenience interface for chimera removal. Two methods to identify chimeras are supported: Identification from pooled sequences (see isBimeraDenovo for details) and identification by consensus across samples (see isBimeraDenovoTable for details). Sequence variants identified as bimeric are removed, and a bimera-free collection of unique sequences is returned.

Usage

```
removeBimeraDenovo(unqs, method = "consensus", ..., verbose = FALSE)
```

Arguments

unqs	(Required). A uniques-vector or any object that can be coerced into one with getUniques. A list of such objects can also be provided.
method	(Optional). Default is "consensus". Only has an effect if a sequence table is provided. If "pooled": The samples in the sequence table are all pooled together for bimera identification (isBimeraDenovo). If "consensus": The samples in a sequence table are independently checked for bimeras, and a consensus decision on each sequence variant is made (isBimeraDenovoTable).

	If "per-sample": The samples in a sequence table are independently checked for bimeras, and sequence variants are removed (zeroed-out) from samples independently (isBimeraDenovo).
...	(Optional). Arguments to be passed to isBimeraDenovo or isBimeraDenovoTable. The documentation of those methods detail the additional algorithmic parameters that can be adjusted.
verbose	(Optional). Default FALSE. Print verbose text output.

Value

A uniques vector, or an object of matching class if a data.frame or sequence table is provided. A list of such objects is returned if a list of input unqs was provided.

See Also

isBimeraDenovoTable, isBimeraDenovo

Examples

```
derep1 = derepFastq(system.file("extdata", "sam1F.fastq.gz", package="dada2"))
dada1 <- dada(derep1, err=tperr1, errorEstimationFunction=loessErrfun, selfConsist=TRUE)
out.nobim <- removeBimeraDenovo(dada1)
out.nobim <- removeBimeraDenovo(dada1$clustering, method="pooled", minFoldParentOverAbundance = 2)
```

---

seqComplexity *Determine if input sequence(s) are low complexity.*

Description

This function calculates the oligonucleotide complexity of input sequences. Complexity is quantified as the Shannon richness of oligonucleotides, which can be thought of as the effective number of oligonucleotides if they were all at equal frequencies. If a window size is provided, the minimum Shannon richness observed over sliding window along the sequence is returned.

Usage

```
seqComplexity(seqs, wordSize = 2, window = NULL, by = 5)
```

Arguments

seqs	(Required). A character vector of A/C/G/T sequences, or any object coercible by getSequences.
wordSize	(Optional). Default 2. The size of the oligonucleotides (or "words" or "kmers") to use.
window	(Optional). Default NULL. The width in nucleotides of the moving window. If NULL the whole sequence is used.
by	(Optional). Default 5. The step size in nucleotides between each moving window tested.

Details

This function can be used to identify potentially artefactual or undesirable low-complexity sequences, or sequences with low-complexity regions, as are sometimes observed in Illumina sequencing runs. When such artefactual sequences are present, a simple plot of the Shannon oligonucleotide richness values returned by this function will typically show a clear bimodal signal.

Value

`numeric`. A vector of minimum olignucleotide complexities for each sequence.

See Also

`oligonucleotideFrequency`

Examples

```
sq.norm <- "TACGGAAGGTCCGGGCGTTATCCGGATTTATTGGGTTTAAAGGGAGCGTAGGCCGGAGATTAAGCGTGTTGTGA"
sq.lowc <- "TCCTTCTTCTCCTCTCTTTCTCCTTCTTTCTTTTTTTTCCCTTTCTCTTCTTCTTTTTCTTCCTTCCTTTTTTC"
sq.part <- "TTTTTCTTCTCCCCCTTCCCCTTTCCTTTTCTCCTTTTTTCCTTTAGTGCAGTTGAGGCAGGCGGAATTCGTGG"
sqs <- c(sq.norm, sq.lowc, sq.part)
seqComplexity(sqs)
seqComplexity(sqs, window=25)
```

`setDadaOpt` *Set DADA options*

Description

setDadaOpt sets the default options used by the dada(...) function for your current session, much like `par` sets the session default plotting parameters. However, all dada options can be set as part of the dada(...) function call itself by including a DADA_OPTION_NAME=VALUE argument.

Usage

```
setDadaOpt(...)
```

Arguments

...	(Required). The DADA options to set, along with their new value.

Details

Sensitivity

OMEGA_A: This parameter sets the threshold for when DADA2 calls unique sequences significantly overabundant, and therefore creates a new partition with that sequence as the center. Default is 1e-40, which is a conservative setting to avoid making false positive inferences, but which comes at the cost of reducing the ability to identify some rare variants.

OMEGA_P: The threshold for unique sequences with prior evidence of existence (see ‘priors‘ argument). Default is 1e-4.

OMEGA_C: The threshold at which unique sequences inferred to contain errors are corrected in the final output. The probability that each unique sequence is generated at its observed abundance from the center of its final partition is evaluated, and compared to OMEGA_C. If that probability is >= OMEGA_C, it is "corrected", i.e. replaced by the partition center sequence. The special value of 0 corresponds to correcting all input sequences, and any value > 1 corresponds to performing no correction on sequences found to contain errors. Default is 1e-40 (same as OMEGA_A).

Alignment

MATCH: The score of a match in the Needleman-Wunsch alignment. Default is 4.

MISMATCH: The score of a mismatch in the Needleman-Wunsch alignment. Default is -5.

GAP_PENALTY: The cost of gaps in the Needleman-Wunsch alignment. Default is -8.

HOMOPOLYMER_GAP_PENALTY: The cost of gaps in homopolymer regions (>=3 repeated bases). Default is NULL, which causes homopolymer gaps to be treated as normal gaps.

BAND_SIZE: When set, banded Needleman-Wunsch alignments are performed. Banding restricts the net cumulative number of insertion of one sequence relative to the other. The default value of BAND_SIZE is 16. If DADA is applied to sequencing technologies with high rates of indels, such as 454 sequencing, the BAND_SIZE parameter should be increased. Setting BAND_SIZE to a negative number turns off banding (i.e. full Needleman-Wunsch).

Sequence Comparison Heuristics

USE_KMERS: If TRUE, a 5-mer distance screen is performed prior to performing each pairwise alignment, and if the 5mer-distance is greater than KDIST_CUTOFF, no alignment is performed. Default is TRUE.

KDIST_CUTOFF: The default value of 0.42 was chosen to screen pairs of sequences that differ by >10%, and was calibrated on Illumina sequenced 16S amplicon data. The assumption is that sequences that differ by such a large amount cannot be linked by amplicon errors (i.e. if you sequence one, you won't get a read of other) and so careful (and costly) alignment is unnecessary.

GAPLESS: If TRUE, the ordered kmer identity between pairs of sequences is compared to their unordered overlap. If equal, the optimal alignment is assumed to be gapless. Default is TRUE. Only relevant if USE_KMERS is TRUE.

GREEDY: The DADA2 algorithm is not greedy, but a very restricted form of greediness can be turned on via this option. If TRUE, unique sequences with reads less than those expected to be generated by resequencing just the central unique in their partition are "locked" to that partition. Modest (~30

New Partition Conditions

MIN_FOLD: The minimum fold-overabundance for sequences to form new partitions. Default value is 1, which means this criteria is ignored.

MIN_HAMMING: The minimum hamming-separation for sequences to form new partitions. Default value is 1, which means this criteria is ignored.

MIN_ABUNDANCE: The minimum abundance for unique sequences form new partitions. Default value is 1, which means this criteria is ignored.

MAX_CLUST: The maximum number of partitions. Once this many partitions have been created, the algorithm terminates regardless of whether the statistical model suggests more real sequence variants exist. If set to 0 this argument is ignored. Default value is 0.

Self Consistency

MAX_CONSIST: The maximum number of steps when selfConsist=TRUE. If convergence is not reached in MAX_CONSIST steps, the algorithm will terminate with a warning message. Default value is 10.

Pseudo-pooling Behavior

PSEUDO_PREVALENCE: When performing pseudo-pooling, all sequence variants found in at least two samples are used as priors for a subsequent round of sample inference. Only relevant if 'pool="pseudo"'. Default is 2.

PSEUDO_ABUNDANCE: When performing pseudo-pooling, all denoised sequence variants with total abundance (over all samples) greater than this are used as priors for a subsequent round of sample inference. Only relevant if 'pool="pseudo"'. Default is Inf (i.e. abundance ignored for this purpose).

Error Model

USE_QUALS: If TRUE, the dada(...) error model takes into account the consensus quality score of the dereplicated unique sequences. If FALSE, quality scores are ignored. Default is TRUE.

Technical

SSE: Controls the level of explicit SSE vectorization for kmer calculations. Default 2. Maintained for development reasons, should have no impact on output.

- 0: No explicit vectorization (but modern compilers will auto-vectorize the code).
- 1: Explicit SSE2.
- 2: Explicit, packed SSE2 using 8-bit integers. Slightly faster than SSE=1.

Value

NULL.

See Also

getDadaOpt

Examples

```
setDadaOpt(OMEGA_A = 1e-20)
setDadaOpt(MATCH=1, MISMATCH=-4, GAP_PENALTY=-6)
setDadaOpt(GREEDY=TRUE, GAPLESS=TRUE)
```

---

show,derep-method *method extensions to show for dada2 objects.*

Description

See the general documentation of show method for expected behavior.

Usage

```
## S4 method for signature 'derep'
show(object)

## S4 method for signature 'dada'
show(object)
```

Arguments

object	Any R object

Value

NULL.

See Also

show

---

tperr1 *An empirical error matrix.*

Description

A dataset containing the error matrix estimated by fitting a piecewise linear model to the errors observed in the mock community featured in Schirmer 2015 (metaID 35).

Format

A numerical matrix with 16 rows and 41 columns. Rows correspond to the 16 transition (eg. A2A, A2C, ...) Columns correspond to consensus quality scores 0 to 40.

---

uniques-vector *The named integer vector format used to represent collections of unique DNA sequences.*

Description

The uniques vector is an `integer` vector that is named by the unique sequence, and valued by the abundance of that sequence. This format is commonly used within the dada2-package, for function inputs and outputs. The getUniques function coerces a variety of input objects into the uniques-vector format, including dada-class and derep-class objects.

See Also

getUniques

uniquesToFasta *Write a uniques vector to a FASTA file*

Description

A wrapper for writeFastq in the ShortRead package. Default output format is compatible with uchime.

Usage

```
uniquesToFasta(unqs, fout, ids = NULL, mode = "w", width = 20000, ...)
```

Arguments

unqs	(Required). A uniques-vector or any object that can be coerced into one with getUniques.
fout	(Required). The file path of the output file.
ids	(Optional). character. Default NULL. A vector of sequence ids, one for each element in unqs. If NULL, a uchime-compatible ID is assigned.
mode	(Optional). Default "w". Passed on to writeFasta indicating the type of file writing mode. Default is "w".
width	(Optional). Default 20000. The number of characters per line in the file. Default is effectively one line per sequence. Passed on to writeFasta.
...	Additional parameters passed on to writeFasta.

Value

NULL.

Examples

```
derep1 = derepFastq(system.file("extdata", "sam1F.fastq.gz", package="dada2"))
outfile <- tempfile(fileext=".fasta")
uniquesToFasta(derep1, outfile)
uniquesToFasta(derep1, outfile, ids=paste0("Sequence", seq(length(getSequences(derep1)))))
```

writeFasta,character-method *Writes a named character vector of DNA sequences to a fasta file. Values are the sequences, and names are used for the id lines.*

Description

Writes a named character vector of DNA sequences to a fasta file. Values are the sequences, and names are used for the id lines.

Usage

```
## S4 method for signature 'character'
writeFasta(object, file, mode = "w", width = 20000L,
  ...)
```

Arguments

object	(Required). A named character vector.
file	(Required). The output file.
mode	(Optional). Default "w". Append with "a".
width	(Optional). Default 20000L. Maximum line length before newline.
...	(Optional). Additonal arguments passed to writeXStringSet.

Value

NULL.

See Also

writeXStringSet

Index