Package ‘dada2’

October 12, 2016

Type Package

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

Description The dada2 package provides ``OTU picking'' functionality, but instead of picking OTUs the DADA2 algorithm exactly infers samples sequences. The dada2 pipeline starts from demultiplexed fastq files, and outputs inferred sample sequences and associated abundances after removing substitution and chimeric errors. Taxonomic classification is also available via a native implementation of the RDP classifier method.

Version 1.0.3

Date 2016-05-22

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)

Imports Biostrings (>= 2.32.1), ggplot2 (>= 1.0), data.table (>= 1.9.4), reshape2 (>= 1.4.1), ShortRead (>= 1.24.0)

Suggests testthat (>= 0.9.1), microbenchmark (>= 1.4.2), BiocStyle, knitr, rmarkdown

VignetteBuilder knitr

biocViews Microbiome, Sequencing, Classification, Metagenomics

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

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

LinkingTo Rcpp

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 5.0.1

NeedsCompilation yes

R topics documented:

dada2-package . . . 3
assignTaxonomy . . . 3
collapseNoMismatch . . . 4
dada . . . 5
dada-class . . . 7
derep-class . . . 8
derepFastq . . . 8
errBalancedF . . . 9
errBalancedR . . . 9
errExtremeF . . . 10
errExtremeR . . . 10
errHmpF . . . 10
errHmpR . . . 11
evaluate_kmers . . . 11
fastqFilter . . . 12
fastqPairedFilter . . . 13
getDadaOpt . . . 16
getSequences . . . 16
getUniques . . . 17
inflateErr . . . 18
isBimera . . . 18
isBimeraDenovo . . . 19
isPhiX . . . 20
isShiftDenovo . . . 21
loessErrfun . . . 22
makeSequenceTable . . . 23
mergePairs . . . 24
mergePairsByID . . . 26
nwalign . . . 29
nwhamming . . . 30
plotComplementarySubstitutions . . . 30
plotErrors . . . 31
plotQualityProfile . . . 32
removeBimeraDenovo . . . 33
setDadaOpt . . . 33
show,derep-method . . . 35
tperr1 . . . 36
uniques-vector . . . 36
uniquesToFasta . . . 36

Index **38**

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 can resolve sequence variants 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 (`fastqFilter`, `fastqPairedFilter`)
- Dereplication (`derepFastq`)
- Sample Inference (`dada`)
- Chimera Removal (`isBimeraDenovo`, `removeBimeraDenovo`)
- Merging of Paired Reads (`mergePairs`)
- Taxonomic Classification (`assignTaxonomy`)

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>

assignTaxonomy *Classifies sequences against reference training dataset.*

Description

assignTaxonomy implements the RDP classifier algorithm in Wang 2007 with kmer size 8 and 100 bootstrap replicates.

Usage

```
assignTaxonomy(seqs, refFasta, minBoot = 50, 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 compresssed. 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.

`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.

Examples

```
## Not run:
 taxa <- assignTaxonomy(dadaF, "gg_13_8_train_set_97.fa.gz")
 taxa <- assignTaxonomy(dadaF, "rdp_train_set_14.fa.gz", minBoot=80)

## End(Not run)
```

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, 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.
`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, ...)
```

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. Each entry must be between 0 and 1.

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. 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.

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.

`...` (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 statiscal 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 clusters of sequencing reads, and the final set of clusters 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 (2015). DADA2: High resolution sample inference from amplicon data. bioRxiv, 024034.
- 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 `message`s 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.

evaluate_kmers — *Generate the kmer-distance and the alignment distance from the given set of sequences.*

Description

Generate the kmer-distance and the alignment distance from the given set of sequences.

Usage

```
evaluate_kmers(seqs, kmer_size, score, gap, band, max_aligns)
```

Arguments

`seqs`	(Required). Character. A vector containing all unique sequences in the data set. Only A/C/G/T allowed.
`kmer_size`	(Required). A `numeric(1)`. The size of the kmer to test (eg. 5-mer).
`score`	(Required). Numeric matrix (4x4). The score matrix used during the alignment. Coerced to integer.
`gap`	(Required). A `numeric(1)` giving the gap penalty for alignment. Coerced to integer.
`band`	(Required). A `numeric(1)` giving the band-size for the NW alignments.
`max_aligns`	(Required). A `numeric(1)` giving the (maximum) number of pairwise alignments to do.

Value

data.frame

Examples

```
derep1 = derepFastq(system.file("extdata", "sam1F.fastq.gz", package="dada2"))
kmerdf <- dada2:::evaluate_kmers(getSequences(derep1), 5, getDadaOpt("SCORE_MATRIX"),
                                  getDadaOpt("GAP_PENALTY"), 16, 1000)
plot(kmerdf$kmer, kmerdf$align)
```

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 and their associated qualities 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, trimLeft = 0, maxN = 0,
  minQ = 0, maxEE = Inf, rm.phix = FALSE, n = 1e+06, compress = TRUE,
  verbose = FALSE, ...)
```

Arguments

`fn` (Required). The path to the input fastq file, or an R connection to that file.

`fout` (Required). The path to the output file, or an R connection to that 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`. The default value of 2 is a special quality score indicating the end of good quality sequence in Illumina 1.8+.

`truncLen` (Optional). Default 0 (no truncation). Truncate reads after `truncLen` bases. Reads shorter than this are discarded. Note that dada currently requires all sequences to be the same length.

`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 FALSE. If TRUE, discard reads that match against the phiX genome, as determined by `isPhiX`.

`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.

`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`.

Details

`fastqFilter` replicates most of the functionality of the fastq_filter command in usearch (http://www.drive5.com/usearch/mar It adds the ability to remove contaminating phiX sequences as part of the filtering process.

Value

NULL.

See Also

`fastqPairedFilter`

`FastqStreamer`

`srFilter`

`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 takes in two input fastq file (can be compressed), filters them based on several user-definable criteria, and outputs those reads which pass the filter in both directions along with their associated qualities 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), trimLeft = c(0, 0), minQ = c(0, 0), maxEE = c(Inf,
  Inf), rm.phix = c(FALSE, FALSE), matchIDs = FALSE, id.sep = "\\s",
  id.field = NULL, n = 1e+06, 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 that follow can be provided as length 1 or length 2 vectors. If a length 1 vector is provided, the same parameter value is used for the forward and reverse sequence files. 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`. The default value of 2 is a special quality score indicating the end of good quality sequence in Illumina 1.8+.

`truncLen` (Optional). Default 0 (no truncation). Truncate reads after `truncLen` bases. Reads shorter than this are discarded. Note that `dada` currently requires all sequences to be the same length.

`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 FALSE. If TRUE, discard reads that match against the phiX genome, as determined by `isPhiX`.

ID MATCHING ARGUMENTS that follow enforce matching between the sequence identification strings in the forward and reverse reads. The function can automatically detect and match ID fields in Illumina format, e.g: EAS139:136:FC706VJ:2:2104:15343:19

`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`).

`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.

`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`.

Details

fastqPairedFilter replicates most of the functionality of the fastq_filter command in usearch (http://www.drive5.com/usearch/m
but only pairs of reads that both pass the filter are retained. An added function is the option to remove contaminating phiX sequences as part of the filtering process.

Value

NULL.

See Also

`fastqFilter`

`FastqStreamer`

`srFilter`

`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)
```

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()
```

getSequences *Get vector of sequences from input object.*

Description

This function extracts the unique 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).

Usage

```
getSequences(object)
```

Arguments

`object`	(Required). The object from which to extract the 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)
```

Arguments

`object`	(Required). The object from which to extract the `uniques-vector`.

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 = TRUE, 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 TRUE. If TRUE, 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 = 1, minParentAbundance = 8,
  allowOneOff = TRUE, minOneOffParentDistance = 4, maxShift = 16,
  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 1. 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 TRUE. If TRUE, 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".

`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=FALSE)
```

---

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 was fount to match 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)
```

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

Description

This function accepts a matrix of observed transitions, with each transition correponding 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 contructs 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 = 20, maxMismatch = 0,
  returnRejects = FALSE, propagateCol = character(0),
  justConcatenate = 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 20. 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.

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

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)]
```

nwalign *Needlman-Wunsch alignment.*

Description

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

Usage

```
nwalign(s1, s2, score = getDadaOpt("SCORE_MATRIX"),
  gap = getDadaOpt("GAP_PENALTY"), homo_gap = NULL, band = -1,
  endsfree = TRUE)
```

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.

`score` (Optional). A 4x4 numeric matrix. Default is getDadaOpt("SCORE_MATRIX"). The match/mismatch used for 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). This 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 free gapping at the ends of sequences.

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)
```

plotComplementarySubstitutions *Plot Substitution Pairs from DADA Result*

Description

This is similar to original DADA article, Figure 6.

Usage

```
plotComplementarySubstitutions(dadaOut, facetByGrp = TRUE)
```

Arguments

`dadaOut`	(Required). A `dada-class` object.
`facetByGrp`	(Optional). Default TRUE. Whether to plot all substitution groups together in one panel or separately on a grid of panels with a linear model fit.

Value

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

Examples

```
derep1 = derepFastq(system.file("extdata", "sam1F.fastq.gz", package="dada2"), verbose = TRUE)
dada1 <- dada(derep1, err = inflateErr(tperr1, 2), selfConsist = TRUE)
plotComplementarySubstitutions(dada1)
```

`plotErrors` *Plot observed error rates after denoising.*

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). A `dada-class` object, or a list of such objects that were pooled to estimate a common set of error rates.
`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.

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.

Usage

```
plotQualityProfile(fl, n = 1e+06)
```

Arguments

fl	(Required). character(1). The file path to the fastq or fastq.gz file.
n	(Optional). Default 1,000,000. The number of records to sample from the fastq file.

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 wrapper around `isBimeraDenovo`. Bimeras identified by isBimeraDenovo are removed, and a bimera-free collection of unique sequences is returned.

Usage

```
removeBimeraDenovo(unqs, ..., 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.
`...`	(Optional). Arguments to be passed to `isBimeraDenovo`.
`verbose`	(Optional). `logical(1)` indicating verbose text output. Default FALSE.

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

`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, minFoldParentOverAbundance = 2, allowOneOff=FALSE)
```

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

The various dada options...

OMEGA_A: This parameter sets the threshold for when DADA2 calls unique sequences significantly overabundant, and therefore creates a new cluster with that sequence as the center. The default value 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.

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. The default is TRUE, however if applying DADA2 to pyrosequenced data it is recommended to set USE_QUALS to FALSE, as quality scores are not informative about substitution error rates in pyrosequencing.

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. TRUE by default.

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.

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 marker genes with high rates of indels, such as the ITS region in fungi, the BAND_SIZE parameter should be increased. Setting BAND_SIZE to a negative number turns off banding (i.e. full Needleman-Wunsch).

SCORE_MATRIX: The score matrix for the Needleman-Wunsch alignment. This is a 4x4 matrix as no ambiguous nucleotides are allowed. Default is nuc44: -4 for mismatches, +5 for matchces.

GAP_PENALTY: The cost of gaps in the Needlman-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.

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

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

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

VERBOSE: If TRUE progress messages from the algorithm are printed. Warning: There is a lot of output. Default is FALSE.

Value

NULL.

See Also

getDadaOpt

Examples

```
setDadaOpt(OMEGA_A = 1e-20)
setDadaOpt(OMEGA_A = 1e-20, VERBOSE = 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.

NULL.

See Also

show

Examples

```
# examples
```

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(getUniques(derep1), outfile)
uniquesToFasta(getUniques(derep1), outfile, ids=paste0("Sequence", seq(length(getUniques(derep1)))))
```

Index

∗Topic **package**
dada2-package, 3

assignTaxonomy, *3*, 3

character, *26*
collapseNoMismatch, 4

dada, *3*, 5, *7, 8*, *12*, *14*, *23–25*, *27*
dada-class, 7
dada2-package, 3
data.frame, *26*
data.table, *27*
derep-class, 8
derepFastq, *3*, *6–8*, 8, *25–27*

errBalancedF, 9
errBalancedR, 9
errExtremeF, 10
errExtremeR, 10
errHmpF, 10
errHmpR, 11
evaluate_kmers, 11

fastqFilter, *3*, 12, *15*, *21*
fastqPairedFilter, *3*, *13*, 13, *21*, *25*
FastqStreamer, *8, 9*, *13*, *15*

getDadaOpt, 16, *35*
getSequences, 16
getUniques, *4*, *16*, 17, *20*, *22, 23*, *33*, *36, 37*
ggplot, *31, 32*
ggsave, *31, 32*
gsub, *26*

id, *26*
inflateErr, 18
isBimera, 18, *19, 20*, *22*
isBimeraDenovo, *3*, *19*, 19, *33*
isPhiX, *13–15*, 20
isShiftDenovo, 21

list, *8*
loess, *22, 23*
loessErrfun, *6*, 22
logical, *26*

makeSequenceTable, *5*, 23
mergePairs, *3*, *14*, 24, *26*
mergePairsByID, 26
message, *9*

nwalign, 29, *30*
nwhamming, 30

plotComplementarySubstitutions, 30
plotErrors, 31
plotQualityProfile, 32
print, *31, 32*

readFastq, *26*
removeBimeraDenovo, *3*, *19, 20*, 33

setDadaOpt, *6, 7*, *16*, 33
show, *35*
show,dada-method (show,derep-method), 35
show,derep-method, 35
srFilter, *13*, *15*
strsplit, *15*

tperr1, 36
trimTails, *13*, *15*

uniques-vector, 36
uniquesToFasta, 36

writeFasta, *37*