Package ‘debar’

December 22, 2019

Type Package

Title A Post-Clustering Denoiser for COI-5P Barcode Data

Version 0.1.0

Author Cameron M. Nugent

Maintainer Cameron M. Nugent <nugentc@uoguelph.ca>

Description The ‘debar’ sequence processing pipeline is designed for denoising high throughput sequencing data for the animal DNA barcode marker cytochrome c oxidase I (COI). The package is designed to detect and correct insertion and deletion errors within sequencer outputs. This is accomplished through comparison of input sequences against a profile hidden Markov model (PHMM) using the Viterbi algorithm (for algorithm details see Durbin et al. 1998, ISBN: 9780521629713). Inserted base pairs are removed and deleted base pairs are accounted for through the introduction of a placeholder character. Since the PHMM is a probabilistic representation of the COI barcode, corrections are not always perfect. For this reason ‘debar’ censors base pairs adjacent to reported indel sites, turning them into placeholder characters (default is 7 base pairs in either direction, this feature can be disabled). Testing has shown that this censorship results in the correct sequence length being restored, and erroneous base pairs being masked the vast majority of the time (>95%).

License GPL-3

Encoding UTF-8

LazyData true

Depends R(>= 3.0.0)

Imports ape, aphid, seqinr, parallel

Suggests knitr, rmarkdown, testthat

VignetteBuilder knitr

RoxygenNote 6.1.1

NeedsCompilation no

Repository CRAN

Date/Publication 2019-12-22 15:50:02 UTC
aa_check

Description

A side product of the framing and adjustment functions is that the reading frame of the sequence is established and translation can be conducted with high confidence. If the adjustment did in fact correct indel errors, the translated sequence should feature no stop codons. If stop codons are present this is grounds for rejection as indel errors (or some other large issue) persists in the sequence.

Usage

aa_check(x, ...)

# S3 method for class 'DNAseq'
aa_check(x, ..., trans_table = 0, frame_offset = 0)

Arguments

x      a ccs_reads class object.
...    additional arguments to be passed between methods.
trans_table  The translation table to use for translating from nucleotides to amino acids. Default is "auto", meaning that the translation table will be inferred from the ccs_reads object’s order.
frame_offset  The offset to the reading frame to be applied for translation. By default the offset is zero, so the first character in the framed sequence is considered the first nucleotide of the first codon. Passing frame_offset = 1 would offset the sequence by one and therefore make the second character in the framed sequence the the first nucleotide of the first codon.

Details
This test has limitations, as any indels late in the DNA sequence may not lead to stop codons existing. Additionally by default censored translation is used by this function when producing the amino acid sequence, so as to eliminate taxonomic bias against organisms with esoteric translation tables. The likelihood of catching errors is increased if the genetic code corresponding to sequences is known.

Value
a class object of code "ccs_reads"

See Also
DNAseq, frame, adjust

Examples
# previously called
ex_data = DNAseq(example_nt_string, name = ’ex1’)  
ex_data = frame(ex_data)  
ex_data = adjust(ex_data)  
# run the aa check on the adjusted sequences
ex_data$aa_seq  
ex_data$stop_codons # Boolean indicating if stop codons in the amino acid sequence

---

Adjust the sequences based on the nt path outputs.

Description
Based on the PHMM path generated by the frame function, the sequence is the adjusted. Adjustments are limited to the 657bp region represented by the PHMM (and the part of the input sequence matching this region). Censorship can be applied around the corrections. This limits the number of indel errors missed by the PHMM correction algorithm, but comes at a cost of lost DNA sequence. The default of 7 is a conservative parameter meant to lead to the coverage of greater than 95
Usage

adjust(x, ...)

## S3 method for class 'DNAseq'
adjust(x, ..., censor_length = 7, added_phred = "*")

Arguments

- **x**: a DNAseq class object.
- **...**: additional arguments to be passed between methods.
- **censor_length**: the number of base pairs in either direction of a PHMM correction to convert to placeholder characters. Default is 7.
- **added_phred**: The phred character to use for characters inserted into the original sequence. Default is ":*":.

Details

If the DNAseq object contains PHRED scores, the PHRED string will be adjusted along with the DNA sequence (corresponding) value removed when a bp removed. The 'added_phred' value indicated the phred character to be added to the string when a placeholder nucleotide is added to the string to account for a deletion. Default is ":*": which indicates a score of 9 (a relatively low quality base).

Value

- a class object of code"ccs_reads"

See Also

- DNAseq
- frame

Examples

```r
# previously called
ex_data = DNAseq(example_nt_string_errors, name = 'error_adj_example')
ex_data = frame(ex_data)
# adjust the sequence with default censor length is 7
ex_data = adjust(ex_data)
ex_data$adjusted_sequence # output is a vector, use outseq to build the string
# with a custom censorship size
ex_data = adjust(ex_data, censor_length = 5)
ex_data$adjusted_sequence # less flanking base pairs turned to placeholders
ex_data$adjustment_count # get a count of the number of adjustments applied
```
**censored_translation**

Censored Translation of a DNA string.

**Description**

Translate a DNA sequence using the censored translation table, this translates codons for which the amino acids is unambiguous across mitochondrial genetic codes across the animal kingdom and does not translate those for which the amino acid varies, but rather outputs a ? in the string.

**Usage**

censored_translation(dna_str, reading_frame = 1)

**Arguments**

- **dna_str**
  - The DNA string to be translated.
- **reading_frame**
  - reading frame = 1 means the first bp in the string is the start of the first codon, can pass 1, 2 or 3. For 2 and 3 the first 1 and 2 bp will be dropped from translation respectively.

**See Also**

- aa_check

**consensus**

Take the list of denoised sequences and obtain the consensus sequence.

**Description**

Take the list of denoised sequences and obtain the consensus sequence.

**Usage**

consensus_sequence(x)

**Arguments**

- **x**
  - The list of adjusted DNA sequences, post censorship and AA correction

**Value**

A string containing the consensus nucleotide sequence.

**See Also**

- denoise_list
Examples

```r
# denoise list of sequences with the k
ex_out = denoise_list(ex_nt_list, keep_flanks=FALSE)

barcode_seq = consensus_sequence(ex_out)
```

Description

debar is an R package designed for the identification and removal of insertion and deletion errors from COI-5P DNA barcode data.

Details

debar is built around the DNAseq object, which takes a COI-5P DNA barcode sequence and optionally its associated name and PHRED quality information as input. The package utilizes a nucleotide profile hidden Markov model (PHMM) for the identification of the COI-5P region of an input sequence and the identification and correction of indel errors from within the COI-5P region of the sequence. Indel corrections are by default applied in a conservative fashion, with subsequent censorship of 7 base pairs in either direction of an indel correction to mask most instances where the exact bp corresponding to the indel was not found exactly. Numerous filtering and double check steps are applied, and the package includes functions for input/output for either fasta or fastq formats.

The denoise pipeline is heavily parameterized so that a user can tailor the denoising execution for their own data structure and goal.

Functions

- **denoise_file** Run the denoise pipeline for each sequence in a specified input file.
- **denoise** Run the denoise pipeline for a specified sequence
- **read_fasta** Read data from a fasta file to a data frame.
- **read_fastq** Read data from a fastq file to a data frame.
- **write_fasta** Write a denoised sequence to the specified fasta file.
- **write_fastq** Write a denoised sequence and associated quality information to the specified fastq file.
- **DNAseq** Builds a DNAseq class object
- **frame** Match a sequence against the COI-5P PHMM using the Viterbi algorithm to establish the reading frame, optional rejection of sequence based on the quality of the match to the PHMM.
- **adjust** Use the PHMM path output corresponding to the sequence to adjust the DNA sequence and remove indels. Optional censorship of sequence around the corrections.
- **aa_check** Translate the adjusted sequence to amino acids and check it for stop codons.
- **outseq** Construct the output data for the given sequence. Optionally can include or exclude sequence data from outside of the COI-5P region (part of sequence that was not denoised).
denoise

Data

- `example_nt_string` An example COI-5P sequence with no errors.
- `example_nt_string_errors` An example COI-5P sequence with two indel errors.

Author(s)

Cameron M. Nugent

denoise Run the denoiser pipeline for a sequence read.

Description

This function runs the complete denoising pipeline for a given input sequence and its corresponding name and phred scores. The default behaviour is set to interface with fastq files (standard output for most sequencers).

Usage

denoise(x, ...)

## Default S3 method:
denoise(x, ..., name = character(), phred = NULL, 
dir_check = TRUE, double_pass = TRUE, min_match = 100, 
max_inserts = 400, censor_length = 7, added_phred = "x", 
adjust_limit = 5, ambig_char = "N", to_file = FALSE, 
keep_flanks = TRUE, keep_phred = TRUE, outformat = "fastq", 
terminate_rejects = TRUE, outfile = NULL, phred_placeholder = ",#", 
aa_check = TRUE, trans_table = 0, frame_offset = 0, 
append = TRUE)

Arguments

<table>
<thead>
<tr>
<th>Argument</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>x</td>
<td>a DNA sequence string.</td>
</tr>
<tr>
<td>...</td>
<td>additional arguments to be passed between methods.</td>
</tr>
<tr>
<td>name</td>
<td>an optional character string. Identifier for the sequence.</td>
</tr>
<tr>
<td>phred</td>
<td>an optional character string. The phred score string corresponding to the nucleotide string. If passed then the input phred scores will be modified along with the nucleotides and carried through to the sequence output. Default = NULL.</td>
</tr>
<tr>
<td>dir_check</td>
<td>A boolean indicating if both the forward and reverse compliments of a sequence should be checked against the PHMM. Default is TRUE.</td>
</tr>
<tr>
<td>double_pass</td>
<td>A boolean indicating if a second pass through the Viterbi algorithm should be conducted for sequences that had leading nucleotides not matching the PHMM. This improves the accurate establishment of reading frame and will reduce false rejections by the amino acid check, but this comes at a cost of additional processing time. Default is TRUE.</td>
</tr>
</tbody>
</table>
The minimum number of sequential matches to the PHMM for a sequence to be denoised. Otherwise flag the sequence as a reject.

The maximum number of sequentation insert states occurring in a sequence (including the flanking regions). If this number is exceeded than the entire read will be discarded if terminate_rejects = TRUE. Default is 400.

the number of base pairs in either direction of a PHMM correction to convert to placeholder characters. Default is 7.

The phred character to use for characters inserted into the original sequence.

the maximum number of corrections that can be applied to a sequence read. If this number is exceeded then the entire read is rejected. Default is 3.

The character to use for ambiguous positions in the sequence that is output to the file. Default is N.

Boolean indicating whether the sequence should be written to a file. Default is TRUE.

Should the regions of the input sequence outside of the barcode region be readded to the denoised sequence prior to outputting to the file. Options are TRUE, FALSE and 'right'. The 'right' option will keep the trailing flank but remove the leading flank. Default is TRUE. False will lead to only the denoised sequence for the 657bp barcode region being output to the file.

Should the original PHRED scores be kept in the output? Default is TRUE.

The format of the output file. Options are fasta or fastq (default) format.

Boolean indicating if analysis of sequences that fail to meet phred quality score or path match thresholds should be terminated early (prior to sequence adjustment and writing to file). Default it true.

The name of the file to output the data to. Default filenames are respectively: denoised.fasta or denoised.fastq.

The character to input for the phred score line. Default is ‘#’. Used with write_fastq and keep_phred == FALSE only.

Boolean indicating whether the amino acid sequence should be generated and assessed for stop codons. Default = TRUE.

The translation table to use for translating from nucleotides to amino acids. Default is 0, meaning that censored translation is performed (amigious codons ignored). Used only when aa_check = TRUE.

The offset to the reading frame to be applied for translation. By default the offset is zero, so the first character in the framed sequence is considered the first nucleotide of the first codon. Passing frame_offset = 1 would offset the sequence by one and therefore make the second character in the framed sequence the first nucleotide of the first codon. Used only when aa_check = TRUE.

Should the denoised sequence be appended to the output file?(TRUE) Or should the sequence overwrite the output file?(FALSE) Default is TRUE.
denoise_file

Details

Since the pipeline is designed for receiving or outputting either fasta or fastq data, this function is heavily parameterized. Note that not all parameters will affect all use cases (i.e. if your output format is to a fasta file, then the phred_placeholder parameter is ignored as this option only pertains to fastq outputs). The user is encouraged to read the vignette for a detailed walkthrough of the denoiser pipeline that will help identify the parameters that relate to their given needs.

Value

a class object of code "DNAseq"

Examples

# Denoise example sequence with default parameters.
ex_data = denoise(example_nt_string_errors,
                    name = 'example_sequence_1',
                    keep_phred = FALSE,
                    to_file = FALSE)

#fastq data from a file
#previously run
fastq_example_file = system.file('extdata/coi_sequel_data_subset.fastq',
                                  package = 'debar')
data = read_fastq(fastq_example_file)
#denoise the first sequence in the file
#use a custom censor length and no amino acid check
dn_dat_1 = denoise(x = data$sequence[[1]],
                    name = data$header[[1]],
                    phred = data$quality[[1]],
                    censor_length = 11,
                    aa_check = FALSE,
                    to_file = FALSE)
Usage
denoise_file(x, ...)

## Default S3 method:
denoise_file(x, ..., outfile = "output.fastq",
informat = "fastq", outformat = "fastq", to_file = TRUE,
log_file = FALSE, keep_rejects = FALSE, multicore = FALSE)

Arguments

x
The name of the file to denoise sequences from.

... additional arguments to be passed to the denoise and input/output functions.

outfile
The name of the file to output sequences to.

informat The format of the file to be denoised. Options are fastq or fasta. Default is fastq.

outformat The format of the output file. Options are fasta or fastq (default) format.

to_file Boolean indicating whether the sequence should be written to a file. Default is TRUE.

log_file Boolean indicating if a log file should be produced. Default is FALSE.

keep_rejects Boolean indicating if the bad reads should be written to a separate file (with the name "rejects_" + outfile). Default is FALSE.

multicore An integer specifying the number of cores over which to multithread the denoising process. Default is FALSE, meaning the process is not multithreaded.

Details

Using this function is optimized by the appropriation of the multicore option, which allows a user to specify a number of cores that the denoising process should be multithreaded across. The more cores available, the faster the denoising of the input data. It should be noted that the multithreading relies on the entire fastq file being read into memory, because of this your machine’s available ram will need to exceed the size of the unzipped fastq file being denoised. If your file size exceeds the available memory you may want to consider splitting the input into several smaller files and denoising them each with this function (this is a fast solution as the multicore option can be used to speed up denoising). Alternatively, you can depoly the ‘denoise’ function in an iterative fashion, reading in/denoising and writing only a single fq entry at a time. This would require a much smaller memory footprint, but would be much slower due to the lack of multithreading.

See Also
denoise
denoise_list

List-to-list denoising of COI barcode sequences.

Description

This function provides a shortcut for running the denoise function on a list of sequences. The to_return option can be used to control whether this function returns a list of sequence strings (default), or a list of DNA seq objects.

Usage

denoise_list(x, to_return = "seq", cores = 1, ...)

Arguments

x A list like object of barcode sequences.
to_return Indicate whether a the function should return a list of sequence ('seq') or the full DNAseq object ('DNAseq'). Default is ('seq')
cores The number of cores across which to thread the denosing. Default is 1.
... additional arguments to pass to the denoise algorithm.

See Also

denoise

denoise_list(x, to_return = "seq", cores = 1, ...)

dir_check

Take an input sequence and align both the forward and reverse complements to the PHMM

Description

The function returns the sequence, DNAbin and Path and score of the optimal orientation. Optimal orientation is determined by the direction with the longer string of consecutive ones in the path.
Usage

dir_check(x)

Arguments

x a DNAseq class object.

DNAseq

Build a DNAseq object from a DNA sequence string.

Description

This can optionally include the DNA sequence’s corresponding PHRED quality values and a se-
quencer identifier as well.

Usage

DNAseq(x = character(), name = character(), phred = NULL)

Arguments

x a nucleotide string. Valid characters within the nucleotide string are: a,t,g,c,-,n. The nucleotid

name an optional character string. Identifier for the sequence.

phred an optional character string. The phred score string corresponding to the nu-

Value

an object of class "coi5p"

Examples

dat = DNAseq(example_nt_string)
# named DNAseq sequence
dat = DNAseq(example_nt_string, name = "example_seq1")
# components in output DNAseq object:
dat$raw
dat$name
**example_nt_string**

*Example coi5p DNA sequence string.*

**Description**

This string of barcode data is used in the package documentation’s examples and within the vignette demonstrating how to use the package.

**Usage**

`example_nt_string`

**Format**

An object of class character of length 1.

---

**example_nt_string_errors**

*Example coi5p DNA sequence string with insertion and deletion errors.*

**Description**

This string of barcode data is used in the package documentation’s examples and within the vignette demonstrating how to use the package.

**Usage**

`example_nt_string_errors`

**Format**

An object of class character of length 1.
ex_nt_list

An example of a list of four coi5p sequences, each containing indel errors.

Description

An example of a list of four coi5p sequences, each containing indel errors.

Usage

ex_nt_list

Format

An object of class list of length 4.

frame

Take a DNAseq object and isolate the COI-5P region.

Description

Raw DNA sequence is taken and compared against the nucleotide profile hidden Markov model (PHMM), generating the statistical information later used to apply corrections to the sequence read. The speed of this function and how it operates are dependent on the parameters chosen. When dir_check == TRUE the function will compare the forward and reverse compliment against the PHMM. Since the PHMM comparison is a computationally intensive process, this option should be set to FALSE if you’re data are all forward reads.

Usage

frame(x, ...)

## S3 method for class 'DNAseq'
frame(x, ..., dir_check = TRUE, double_pass = TRUE,
       min_match = 100, max_inserts = 400, terminate_rejects = TRUE)

Arguments

x a DNAseq class object.
...
additional arguments to be passed between methods.
dir_check A boolean indicating if both the forward and reverse compliments of a sequence should be checked against the PHMM. Default is TRUE.
The `outseq` function provides the final denoised output sequence for a read. It is used after the PHMM-based filtering and alignment processes to produce the final sequence after any necessary corrections and denoising. The function operates on DNA sequences and is tailored for handling nucleotide data.

### Description

The `outseq` function is particularly valuable in scenarios where high-quality nucleotide sequences are essential. It allows for the retrieval of sequences that have been processed through various computational stages, including alignment and filtering, ensuring that the final output is as accurate and reliable as possible. This function is crucial for applications in bioinformatics, genetics, and molecular biology, where the integrity and accuracy of the nucleotide sequences are paramount for further analysis and research.

### Examples

To illustrate the usage of `outseq`, consider the following example:

```r
# previously called
hi_phred = paste0(rep("~~~", 217), collapse="")
dat1 = DNAseq(example_nt_string_errors, name = "err_seq1", phred = hi_phred)
dat1= frame(dat1)
dat1$frame_dat # a labelled list with framing information is produced
dat1$reject == FALSE # the match criteria were met, not labelled for rejection
dat1$data$path # one can call and view the path diagram produced by the PHMM comparison.
```

This example demonstrates how `outseq` can be used to process a sequence, frame it, and access its denoising and filtering data. The `dat1$frame_dat` object provides a labelled list with framing information, which is useful for understanding how the sequence was processed and where any denoising or filtering occurred. The `dat1$reject` check confirms that the match criteria were met, avoiding false rejections. Finally, accessing `dat1$data$path` allows for visualization of the path diagram produced by the PHMM comparison, offering insights into the sequence's alignment and filtering processes.
Usage

outseq(x, ...)

## S3 method for class 'DNAseq'
outseq(x, ..., keep_flanks = TRUE, ambig_char = "N",
       adjust_limit = 5)

Arguments

x a DNAseq class object.
...
keep_flanks Should the regions of the input sequence outside of the barcode region be readded
to the denoised sequence prior to outputting to the file. Options are TRUE, FALSE and 'right'. The 'right' option will keep the trailing flank but remove the leading flank. Default is TRUE. False will lead to only the denoised sequence for the 657bp barcode region being output to the file.
ambig_char The character to use for ambiguous positions in the sequence.
adjust_limit the maximum number of corrections that can be applied to a sequence read. If this number is exceeded then the entire read is masked with ambiguous characters. Default is 5.

Value

a class object of code"ccs_reads"

See Also

DNAseq
frame
adjust

Examples

#previously run
excess_string = paste0("CCCCCC", example_nt_string_errors,
                        "CCCCCCCC", collapse="")
ex_data = DNAseq(excess_string, name = 'ex1')
ex_data = frame(ex_data)
ex_data = adjust(ex_data)
#build output sequence with trimmed edges
ex_data = outseq(ex_data, keep_flanks = TRUE)
ex_data$outseq #view the output sequence, edges were reattached
#you will avoid data loss on edge of sequence, but errors in edge, or
#off target sequence will be present in the output
#
#build output sequence with only the COI-5P region
ex_data = outseq(ex_data, keep_flanks = FALSE)
ex_data$outseq #view the output sequence
#Ns added to the front to buffer trimmed region
#Note some sequence is lost due to the strange
#path match that occurs at the front of the sequence.
ex_data$data$path

---

**read_fasta**

*Read in raw data from a fasta file.*

**Description**

Read in raw data from a fasta file.

**Usage**

```r
read_fasta(x)
```

**Arguments**

- `x`: The name of the fasta file to read data from.

**Examples**

```r
fasta_example_file = system.file('extdata/coi_sequel_data_subset.fastq', package = 'debar')
data = read_fasta(fasta_example_file)
```

---

**read_fastq**

*Read in raw data from a fastq file.*

**Description**

Read in raw data from a fastq file.

**Usage**

```r
read_fastq(x, keep_quality = TRUE)
```

**Arguments**

- `x`: The name of the fastq file to read data from.
- `keep_quality`: Boolean indicating if the Phred quality scores should be retained in the output dataframe. Default is TRUE.
Examples

# read in an unzipped fastq file
fastq_example_file = system.file("extdata/coi_sequel_data_subset.fastq",
    package = "debar")
data = read_fastq(fastq_example_file)

# read in a gzipped fastq file and do not keep the phred scores
gz_fastq_example_file = system.file("extdata/coi_sequel_data_subset.fastq",
    package = "debar")
data2 = read_fastq(gz_fastq_example_file, keep_quality = FALSE)

write_fasta

Output the denoised consensus sequence to a fasta file.

Description

Output the denoised consensus sequence to a fasta file.

Usage

write_fasta(x, ...)

## S3 method for class 'DNAseq'
write_fasta(x, ..., outfile = "denoised.fasta",
    append = TRUE)

Arguments

x      a DNAseq class object.
...
    additional arguments to be passed between methods.
outfile The name of the file to output the data to. Default is "denoised.fasta".
append Should the ccs consensus sequence be appended to the output file?(TRUE) Or
         overwrite the file?(FALSE) Default is TRUE.

Value

a class object of code"DNAseq"

See Also

dnaseq
frame
adjust
write_fastq

Output the denoised sequence to a fastq format with placeholder phred scores.

Description

Output the denoised sequence to a fastq format with placeholder phred scores.

Usage

write_fastq(x, ...)

## S3 method for class 'DNaseq'
write_fastq(x, ..., outfile = "denoised.fastq", 
append = TRUE, keep_phred = TRUE, phred_placeholder = ")

Arguments

x a DNaseq class object.

... additional arguments to be passed between methods.

outfile The name of the file to output the data to. Default is "denoised.fasta".

append Should the ccs consensus sequence be appended to the output file?(TRUE) Or overwrite the file?(FALSE) Default is TRUE.

keep_phred Should the original PHRED scores be kept in the output? Default is TRUE.

phred_placeholder The character to input for the phred score line. Default is ").

Value

a class object of code"DNaseq"

See Also

DNaseq
frame
adjust
Index

*Topic datasets
  ex_nt_list, 14
  example_nt_string, 13
  example_nt_string_errors, 13

aa_check, 2, 5, 6
adjust, 3, 3, 6, 16, 18, 19

censored_translation, 5
consensus, 5
consensus_sequence (consensus), 5

debar, 6
debar-package (debar), 6
denoise, 6, 7, 10, 11
denoise_file, 6, 9
denoise_list, 5, 11
dir_check, 11
DNAseq, 3, 4, 6, 12, 15, 16, 18, 19

ex_nt_list, 14
example_nt_string, 7, 13
example_nt_string_errors, 7, 13

frame, 3, 4, 6, 14, 16, 18, 19

outseq, 6, 15

read_fasta, 6, 17
read_fastq, 6, 17

write_fasta, 6, 18
write_fastq, 6, 19