Package ‘FastqCleaner’

May 3, 2024

Type Package
Title A Shiny Application for Quality Control, Filtering and Trimming of FASTQ Files
Version 1.22.0
Date 2022-05-01
Description An interactive web application for quality control, filtering and trimming of FASTQ files. This user-friendly tool combines a pipeline for data processing based on Biostrings and ShortRead infrastructure, with a cutting-edge visual environment. Single-Read and Paired-End files can be locally processed. Diagnostic interactive plots (CG content, per-base sequence quality, etc.) are provided for both the input and output files.
License MIT + file LICENSE
LazyData TRUE
Imports methods, shiny, stats, IRanges, Biostrings, ShortRead, DT, S4Vectors, graphics, htmltools, shinyBS, Rcpp (>= 0.12.12)
Suggests BiocStyle, testthat, knitr, rmarkdown
LinkingTo Rcpp
Collate 'roxygen.auxiliar.R' 'auxiliar.R' 'matching.R'
 'server_functions.R' 'in_filter.R' 'seq_filter.R'
 'complex_filter.R' 'adapter_filter.R' 'launch_fqc.R'
 'length_filter.R' 'fixed_filter.R' 'trim3q_filter.R'
 'unique_filter.R' 'plotObjects.R' 'qmean_filter.R' 'simulate.R'
 'RcppExports.R'
biocViews QualityControl,Sequencing,Software,SangerSeq,SequenceMatching
VignetteBuilder knitr
Encoding UTF-8
RoxygenNote 7.1.2
git_url https://git.bioconductor.org/packages/FastqCleaner
git_branch RELEASE_3_19
git_last_commit 4a8e37c
Contents

adapter_filter .................................................. 3
asc2int ............................................................ 5
check_encoding ..................................................... 5
check_onclick ...................................................... 6
complex_filter .................................................... 6
create_cleanfunction ............................................. 8
create_uniform_width ............................................ 8
cutRseq ........................................................... 9
fixed_filter ....................................................... 11
inject_letter_random ............................................. 12
int2asc ............................................................. 13
isNaturalNumber .................................................. 13
launch_fqc ........................................................ 14
length_filter ...................................................... 14
messageFun ........................................................ 15
myPlot ............................................................ 16
n_filter ............................................................ 16
outputClean ....................................................... 17
plotA ............................................................... 18
plotB ............................................................... 18
plotC ............................................................... 18
plotD ............................................................... 19
plotE ............................................................... 19
plotF ............................................................... 19
plotG ............................................................... 20
plotH ............................................................... 20
plotI ............................................................... 20
plotJ ............................................................... 21
plotObjects ....................................................... 21
processingFunction .............................................. 22
qmean_filter ...................................................... 22
random_length .................................................... 23
random_qual ....................................................... 25
random_seq ....................................................... 26
seq_filter ......................................................... 27
seq_names ........................................................ 28
adapter_filter

Remove full and partial adapters from a ShortReadQ object

Description

This program can remove adapters and partial adapters from 3' and 5', using the functions trimLRPatterns. The program extends the methodology of the trimLRPatterns function of Biostrings, being also capable of removing adapters present within reads and with other additional options (e.g., threshold of minimum number of bases for trimming). For a given position in the read, the two Biostrings functions return TRUE when a match is present between a substring of the read and the adapter. As trimLRPatterns, adapter_filter also selects region and goes up to the end of the sequence in the corresponding flank as the best match. The default error rate is 0.2. If several valid matches are found, the function removes the largest subsequence. Adapters can be anchored or not. When indels are allowed, the second method uses the 'edit distance' between the subsequences and the adapter.

Usage

adapter_filter(
  input,
  Lpattern = "",
  Rpattern = "",
  rc.L = FALSE,
  rc.R = FALSE,
  first = c("R", "L"),
  with_indels = FALSE,
  error_rate = 0.2,
  anchored = TRUE,
  fixed = "subject",
  remove_zero = TRUE,
  checks = TRUE,
  min_match_flank = 3L,
  ...
)

Arguments

input ShortReadQ object
Lpattern 5' pattern (character or DNAString object)
Rpattern 3' pattern (character or DNAString object)
rc.L Reverse complement Lpattern? default FALSE
rc.R Reverse complement Rpattern? default FALSE
adapter_filter

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>first</td>
<td>trim first right (’R’) or left (’L’) side of sequences when both Lpattern and Rpattern are passed</td>
</tr>
<tr>
<td>with_indels</td>
<td>Allow indels? This feature is available only when the error_rate is not null</td>
</tr>
<tr>
<td>error_rate</td>
<td>Error rate (value in the range [0, 1] The error rate is the proportion of mismatches allowed between the adapter and the aligned portion of the subject. For a given adapter A, the number of allowed mismatches between each subsequence s of A and the subject is computed as: error_rate * L_s, where L_s is the length of the subsequence s</td>
</tr>
<tr>
<td>anchored</td>
<td>Adapter or partial adapter within sequence (anchored = FALSE, default) or only in 3’ and 5’ terminals? (anchored = TRUE)</td>
</tr>
<tr>
<td>fixed</td>
<td>Parameter passed to trimLRPatterns Default ’subject’, ambiguities in the pattern only are interpreted as wildcard. See the argument fixed in trimLRPatterns</td>
</tr>
<tr>
<td>remove_zero</td>
<td>Remove zero-length sequences? Default TRUE</td>
</tr>
<tr>
<td>checks</td>
<td>Perform checks? Default TRUE</td>
</tr>
<tr>
<td>min_match_flank</td>
<td>Do not trim in flanks of the subject, if a match has min_match_flank of less length. Default 1L (only trim with &gt;=2 coincidences in a flank match)</td>
</tr>
<tr>
<td>...</td>
<td>additional parameters passed to trimLRPatterns</td>
</tr>
</tbody>
</table>

Value

Edited DNAString or DNAStringSet object
Filtered ShortReadQ object

Author(s)

Leandro Roser <learoser@gmail.com>

Examples

```r
require('Biostrings')
require('ShortRead')

# create 6 sequences of width 43
set.seed(10)
input <- random_seq(6, 43)

# add adapter in 3'
adapter <- "ATCGACT"
input <- paste0(input, as.character(DNAString(adapter)))
input <- DNAStringSet(input)

# create qualities of width 50
set.seed(10)
input_q <- random_qual(c(30,40), slength = 6, swidth = 50, encod = 'Sanger')
```
# create names
input_names <- seq_names(length(input))

# create ShortReadQ object
my_read <- ShortReadQ(sread = input, quality = input_q, id = input_names)

# trim adapter
filtered <- adapter_filter(my_read, Rpattern = adapter)

# look at the filtered sequences
sread(filtered)

---

**asc2int**

*ASCII to integer*

**Description**

ASCII to integer

**Usage**

asc2int(x)

**Value**

Integer

---

**check_encoding**

*Check quality encoding*

**Description**

Check quality encoding

**Usage**

check_encoding(x = NULL, custom = NULL)

**Arguments**

- **x**: Quality values
- **custom**: custom encoding from the following:
  - 'Sanger' ——–> expected range: [0, 40]
  - 'Illumina1.8' ——–> expected range: [0, 41]
  - 'Illumina1.5' ——–> expected range: [0, 40]
  - 'Illumina1.3' ——–> expected range: [3, 40]
  - 'Solexa' ——–> expected range: [-5, 40]
complex_filter

Value
List with encoding information

Author(s)
Leandro Roser <learoser@gmail.com>

Examples
require(Biostrings)

x <- list(PhredQuality(0:40), SolexaQuality(-5:40), IlluminaQuality(3:40))
x <- lapply(x, function(i)utf8ToInt(as.character(i)[1]))
lapply(x, check_encoding)

SolexaQuality(0:40)
IlluminaQuality(0:40)

______________________________
check_onclick_  check onclick
______________________________

Description
Function to put a tickmark on click

Usage
check_onclick_(.menu_react, .butt_number, my_envir)

Value
Change value of reactive output, without return

______________________________
complex_filter  Remove sequences with low complexity
______________________________

Description
The program removes low complexity sequences, computing the entropy with the observed frequency of dinucleotides.

Usage
complex_filter(input, threshold = 0.5, referenceEntropy = 3.908135)
complex_filter

Arguments

input ShortReadQ object
threshold A threshold value computed as the relation of the H of the sequences and the
reference H. Default is 0.5
referenceEntropy Reference entropy. By default, the program uses a value of 3.908, that corre-
sponds to the entropy of the human genome in bits

Value

Filtered ShortReadQ object

Author(s)

Leandro Roser <LEARoser@gmail.com>

Examples

require('Biostrings')
require('ShortRead')

# create sequences of different width
set.seed(10)
input <- lapply(c(0, 6, 10, 16, 20, 26, 30, 36, 40),
function(x) random_seq(1, x))

# create repetitive 'CG' sequences with length adequate
# for a total length:
# input + CG = 40
set.seed(10)
CG <- lapply(c(20, 17, 15, 12, 10, 7, 5, 2, 0),
function(x) paste(rep('CG', x), collapse = ''))

# concatenate input and CG
input <- mapply('paste', input, CG, sep = '')
input <- DNAStringSet(input)

# plot relative entropy (E, Shannon 1948)
freq <- dinucleotideFrequency(input)
freq <- freq /rowSums(freq)
H <- -rowSums(freq * log2(freq), na.rm = TRUE)
H_max <- 3.908135 # max entropy
plot(H/H_max, type='b', xlab = 'Sequence', ylab = 'E')

# create qualities of width 40
set.seed(10)
input_q <- random_qual(c(30,40), slength = 9, swidth = 40,
  encod = 'Sanger')

# create names
input_names <- seq_names(9)

# create ShortReadQ object
my_read <- ShortReadQ(sread = input, quality = input_q, id = input_names)

# apply the filter
filtered <- complex_filter(my_read)

# look at the filtered sequences
sread(filtered)

create_cleanfunction_    create_cleanfunction_

Description
Create a function to process FASTQ files in function of the Shiny parameters selected by the user

Usage
create_cleanfunction_(my_envir, .which_read = c("FORWARD", "REVERSE"))

Value
Function with selected cleaning operations

create_uniform_width    Create fastq/sequences/qualities with uniform width

Description
Create fastq/sequences/qualities with uniform width

Usage
create_uniform_width(input, type = c("fastq", "sequence", "quality"))

Arguments
input  input to edit
type   type of the input: 'fastq' (ShortReadQ), 'sequence' (DNAStringSet), 'quality' (BStringset)
cutRseq

Value

ShortReadQ object or character vector with sequences or qualities, with uniform width (padded with Ns or }

Description

This set of programs are internal, and the function adapter_filter is recommended for trimming. The programs can remove adapters and partial adapters from 3' and 5'. The adapters can be anchored or not. When indels are allowed, the error rate consists in the edit distance. IUPAC symbols are allowed. The methods use the trimLRPatterns function of the Biostrings package, with some additions to take into account e.g., partial adaptors. IUPAC symbols are allowed in all the cases. The present function also removes partial adapters, without the need of additional steps (for example, creating a padded adapter with 'Ns', etc). A similar result to the output of trimLRPatterns can be obtained with the option anchored = TRUE. When several matches are found, the function removes the subsequence that starts in the first match when cutRseq is used, or ends in the last match when cutLseq is used.

Usage

cutRseq(
  subject,
  Rpattern,
  with.indels = FALSE,
  fixed = "subject",
  error_rate = 0.2,
  anchored = TRUE,
  ranges = FALSE,
  checks = TRUE,
  min_match_flank = 2L,
  ...
)

cutLseq(
  subject,
  Lpattern,
  with.indels = FALSE,
  fixed = "subject",
  error_rate = 0.2,
  anchored = TRUE,
  ranges = FALSE,
  checks = TRUE,
  min_match_flank = 3L,
  ...
)
Arguments

- **subject**: DNAString or DNAStringSet object
- **Rpattern**: 3’ pattern, DNAString object
- **with.indels**: Allow indels?
- **fixed**: Parameter passed to trimLRPatterns. Default 'subject', ambiguities in the pattern only are interpreted as wildcard. See the argument fixed in trimLRPatterns.
- **error_rate**: Error rate (value in [0, 1]). The error rate is the proportion of mismatches allowed between the adapter and the aligned portion of the subject. For a given adapter A, the number of allowed mismatches between each subsequence s of A and the subject is computed as: error_rate * L_s, where L_s is the length of the subsequence s.
- **anchored**: Can the adapter or partial adapter be within the sequence? (anchored = FALSE) or only in the terminal regions of the sequence? (anchored = TRUE). Default TRUE (trim only flanking regions)
- **ranges**: Return ranges? Default FALSE
- **checks**: Perform internal checks? Default TRUE
- **min_match_flank**: Do not trim in flanks of the subject, if a match has min_match_flank of less length. Default 1L (only trim with >=2 coincidences in a flank match)
- **...**: additional parameters passed to trimLRPatterns
- **Lpattern**: 5’ pattern, DNAString object

Value

Edited DNAString or DNAStringSet object

Author(s)

Leandro Roser <learoser@gmail.com>

Examples

library(Biostrings)

subject <- DNAStringSet(c('ATCATGCCATCATGAT',
'CATGATATTA', 'TCATG', 'AAAAA', 'AGGTCATG'))

Lpattern <- Rpattern <- 'TCATG'

FastqCleaner:::cutLseq(subject, Lpattern)
FastqCleaner:::cutLseq(subject, Lpattern, ranges = TRUE)
FastqCleaner:::cutRseq(subject, Rpattern)

FastqCleaner:::cutLseq(subject, Lpattern, anchored = FALSE)
FastqCleaner:::cutLseq(subject, Lpattern, error_rate = 0.2)
FastqCleaner:::cutLseq(subject, Lpattern, error_rate = 0.2,
fixed_filter

with.indels = TRUE)

fixed_filter  Remove a fixed number of bases of a ShortReadQ object from 3’ or 5’

Description

The program removes a given number of bases from the 3’ or 5’ regions of the sequences contained in a ShortReadQ object

Usage

fixed_filter(input, trim3 = NA, trim5 = NA)

Arguments

input ShortReadQ object
trim3 Number of bases to remove from 3’
trim5 Number of bases to remove from 5’

Value

Filtered ShortReadQ object

Author(s)

Leandro Roser <learoser@gmail.com>

Examples

require('Biostrings')
require('ShortRead')

# create 6 sequences of width 20
set.seed(10)
input <- random_seq(6, 20)

# create qualities of width 20
set.seed(10)
input_q <- random_qual(c(30,40), slength = 6, swidth = 20, encod = 'Sanger')

# create names
input_names <- seq_names(6)
# create ShortReadQ object
my_read <- ShortReadQ(sread = input, quality = input_q, id = input_names)

# apply the filter
filtered3 <- fixed_filter(my_read, trim5 = 5)
filtered5 <- fixed_filter(my_read, trim3 = 5)
filtered3and5 <- fixed_filter(my_read, trim3 = 10, trim5 = 5)

# look at the trimmed sequences
sread(filtered3)
sread(filtered5)
sread(filtered3and5)

---

**inject_letter_random**  
Inject a letter in a set of sequences at random positions

**Description**
Inject a letter in a set of sequences at random positions

**Usage**

```r
inject_letter_random(
  my_seq,
  how_many_seqs = NULL,
  how_many_letters = NULL,
  letter = "N"
)
```

**Arguments**

- **my_seq**: character vector with sequences to inject
- **how_many_seqs**: How many sequences pick to inject Ns. An interval \([\text{min}\_s, \text{max}\_s]\) with \(\text{min}\_s\) minimum and \(\text{max}\_s\) maximum sequences can be passed. In this case, a value is picked from the interval. If NULL, a random value within the interval \([1, \text{length(my_seq)}]\) is picked.
- **how_many_letters**: How many times inject the letter in the i sequences that are going to be injected. An interval \([\text{min}\_i, \text{max}\_i]\) can be passed. In this case, a value is randomly picked for each sequence i. This value represents the number of times that the letter will be injected in the sequence i. If NULL, a random value within the interval \([1, \text{width(my_seq[i])}]\) is picked for each sequence i.
- **letter**: Letter to inject. Default: ‘N’
Value

character vector

Author(s)

Leandro Roser <learoser@gmail.com>

Examples

# For reproducible examples, make a call to set.seed before
# running each random function

set.seed(10)
s <- random_seq(slengt = 10, swidt = 20)

set.seed(10)
s <- inject_letter_random(s, how_many_seqs = 1:30, how_many= 2:10)
launch_fqc  

Launch FastqCleaner application

Description
Launch FastqCleaner application

Usage
launch_fqc(launch.browser = TRUE, ...)

Arguments
launch.browser  Launch in browser? Default TRUE
...  Additional parameters passed to runApp

Value
Launch the application, without return value

Author(s)
Leandro Roser <learoser@gmail.com>

Examples
# Uncomment and paste in te console to launch the application:
# launch_fqc()

NULL

length_filter  

Filter sequences of a FASTQ file by length

Description
The program removes from a ShortReadQ object those sequences with a length lower than rm.min or/and higher than rm.max

Usage
length_filter(input, rm.min = NA, rm.max = NA)
messageFun_

Arguments

input ShortReadQ object
rm.min Threshold value for the minimum number of bases
rm.max Threshold value for the maximum number of bases

Value

Filtered ShortReadQ object

Author(s)

Leandro Roser <learoser@gmail.com>

Examples

```r
require("Biostrings")
require("ShortRead")

# create ShortReadQ object width widths between 1 and 100
set.seed(10)
input <- random_length(100, widths = 1:100)

# apply the filter, removing sequences length < 10 or length > 80
filtered <- length_filter(input, rm.min = 10, rm.max = 80)

# look at the filtered sequences
sread(filtered)
```

Description

messageFun_

Usage

messageFun_(.who, .chunk, .which_read, my_envir)

Value

Changes the state of reactive vector, without return
myPlot

Description

Construction of diagnostic plots. The function depends of the values created by plotObject

Usage

myPlot(isPaired, location, sampleSize, kmerLength, theFile, maxFreq)

Value

List with Highcharts plots

n_filter

Remove sequences with non-identified bases (Ns) from a ShortReadQ object

Description

This program is a wrapper to nFilter. It removes the sequences with a number of N’s above a threshold value ‘rm.N’. All the sequences with a number of N > rm.N (N >= rm.N) will be removed

Usage

n_filter(input, rm.N)

Arguments

input ShortReadQ object

rm.N Threshold value of N’s to remove a sequence from the output (sequences with number of Ns > threshold are removed) For example, if rm.N is 3, all the sequences with a number of Ns > 3 (Ns >= 4) will be removed

Value

Filtered ShortReadQ object

Author(s)

Leandro Roser <learoser@gmail.com>
Examples

```r
require('Biostrings')
require('ShortRead')

# create 6 sequences of width 20
set.seed(10)
input <- random_seq(50, 20)

# inject N's
set.seed(10)
input <- inject_letter_random(input, how_many_seqs = 1:30,
how_many = 1:10)
input <- DNAStringSet(input)

# watch the N's frequency
hist(letterFrequency(input, 'N'), breaks = 0:10,
main = 'Ns Frequency', xlab = '# Ns')

# create qualities of width 20
set.seed(10)
input_q <- random_qual(50, 20)

# create names
input_names <- seq_names(50)

# create ShortReadQ object
my_read <- ShortReadQ(sread = input, quality = input_q, id = input_names)

# apply the filter
filtered <- n_filter(my_read, rm.N = 3)

# watch the filtered sequences
sread(filtered)

# watch the N's frequency
hist(letterFrequency(sread(filtered), 'N'),
main = 'Ns distribution', xlab = '')
```

Description

`outputClean_` (`.myFile`, `.lengthWidthVec`, `my_envir`)

Usage

`outputClean_`
plotC

Value
Vector with chunks length and width information

plotA

Description
plotA

Usage
plotA(x, nplots = 1, theFile = c("input", "output"), sampleSize)

Value
Per cycle quality plot

plotB

Description
plotB

Usage
plotB(x, nplots = 1, theFile = c("input", "output"), sampleSize)

Value
Per cycle mean base quality plot

plotC

Description
plotC

Usage
plotC(x, nplots = 1, theFile = c("input", "output"), sampleSize)

Value
Mean quality of reads distribution plot
**plotD**

---

**plotD**

**Description**

plotD

**Usage**

```r
plotD(x, nplots = 1, theFile = c("input", "output"), sampleSize)
```

**Value**

percent of reads with quality > threshold plot

---

**plotE**

---

**plotE**

**Description**

plotE

**Usage**

```r
plotE(x, nplots = 1, theFile = c("input", "output"), sampleSize)
```

**Value**

Per cycle base proportion plot

---

**plotF**

---

**plotF**

**Description**

plotF

**Usage**

```r
plotF(x, nplots = 1, theFile = c("input", "output"), sampleSize)
```

**Value**

Per cycle base proportion plot (lineplot)
**Description**

plotG

**Usage**

plotG(x, nplots = 1, theFile = c("input", "output"), sampleSize)

**Value**

CG content distribution plot

---

**Description**

plotH

**Usage**

plotH(x, nplots = 1, theFile = c("input", "output"), sampleSize)

**Value**

Read length distribution

---

**Description**

plotI

**Usage**

plotI(x, nplots = 1, theFile = c("input", "output"), sampleSize)

**Value**

Read occurrence distribution plot
plotJ

Description

plotJ

Usage

plotJ(x, nplots = 1, theFile = c("input", "output"), sampleSize)

Value

Relative kmer diversity plot

plotObjects

plotObjects Create the information required to construct the plots. This is the input of myplot, which uses the values created for this function to construct the plots

Description

plotObjects Create the information required to construct the plots. This is the input of myplot, which uses the values created for this function to construct the plots

Usage

plotObjects(fq, klength, basename, maxFreq, sampleSize)

Value

List with information to construct the diagnostic plots
Description

This function is the core of the application. It is used for the program to process the FASTQ file/s in the environment of the Shiny app. Note that this program makes a call to create_cleanfunction

Usage

processingFunction_(my_envir)

Value

Processes the input FASTQ file, without return

qmean_filter

Filter sequences by their average quality

Description

The program removes the sequences with a quality lower the 'minq' threshold

Usage

qmean_filter(input, minq, q_format = NULL, check.encod = TRUE)

Arguments

input ShortReadQ object
minq Quality threshold
q_format Quality format used for the file, as returned by check.encoding
check.encod Check the encoding of the sequence? This argument is incompatible with q_format

Value

Filtered ShortReadQ object

Author(s)

Leandro Roser <learoser@gmail.com>
Examples

```r
require(ShortRead)

set.seed(10)
# create 30 sequences of width 20
input <- random_seq(30, 20)

# create qualities of width 20
## high quality (15 sequences)
set.seed(10)
my_qual <- random_qual(c(30,40), slength = 15, swidth = 20,
                         encod = 'Sanger')
## low quality (15 sequences)
set.seed(10)
my_qual_2 <- random_qual(c(5,30), slength = 15, swidth = 20,
                          encod = 'Sanger')

# concatenate vectors
input_q<- c(my_qual, my_qual_2)

# create names
input_names <- seq_names(30)

# create ShortReadQ object
my_read <- ShortReadQ(sread = input, quality = input_q, id = input_names)

# watch the average qualities
alphabetScore(my_read) / width(my_read)

# apply the filter
filtered <- qmean_filter(my_read, minq = 30)

# watch the average qualities
alphabetScore(my_read) / width(my_read)

# watch the filtered sequences
sread(filtered)
```

random_length

Create a named object with random sequences and qualities

Description

Create a ShortReadQ object with random sequences and qualities
Usage

random_length(
  n,
  widths,
  random_widths = TRUE,
  replace = TRUE,
  len_prob = NULL,
  seq_prob = c(0.25, 0.25, 0.25, 0.25),
  q_prob = NULL,
  nuc = c("DNA", "RNA"),
  qual = NULL,
  encod = c("Sanger", "Illumina1.8", "Illumina1.5", "Illumina1.3", "Solexa"),
  base_name = "s",
  sep = "_"
)

Arguments

n number of sequences
widths width of the sequences
random_widths width must be picked at random from the passed parameter 'widths', considering the value as an interval where any integer can be picked. Default TRUE. Otherwise, widths are picked only from the vector passed.
replace sample widths with replacement? Default TRUE.
len_prob vector widths with replacement? Default TRUE.
seq_prob vector with probabilities for each width value. Default NULL (equiprobability)
q_prob a vector of four probabilities values to set the frequency of the nucleotides 'A', 'C', 'G', 'T', for DNA, or 'A', 'C', 'G', 'U', for RNA. For example = c(0.25, 0.25, 0.5, 0). Default is = c(0.25, 0.25, 0.25, 0.25) (equiprobability for the 4 bases). If the sum of the probabilities is > 1, the values will be normalized to the range [0, 1].
q_prob a vector of range = range(qual), with probabilities to set the frequency of each quality value. Default is equiprobability. If the sum of the probabilities is > 1, the values will be normalized to the range [0, 1].
nuc create sequences of DNA (nucleotides = c('A', 'C', 'G', 'T')) or RNA (nucleotides = c('A', 'C', 'G', 'U')). Default: 'DNA'
qual quality range for the sequences. It must be a range included in the selected encoding:
'Sanger' = [0, 40]
'Illumina1.8' = [0, 41]
'Illumina1.5' = [0, 40]
'Illumina1.3' = [3, 40]
'Solexa' = [-5, 40]
example: for a range from 20 to 30 in Sanger encoding, pass the argument = c(20, 30)
encod sequence encoding
random_qual

base_name  Base name for strings
sep        Character separating base names and the read number. Default: '_'

Value

ShortReadQ object

Author(s)

Leandro Roser <learoser@gmail.com>

Examples

# For reproducible examples, make a call to set.seed before
# running each random function

set.seed(10)
s1 <- random_seq(slength = 10, swidth = 20)
s1

set.seed(10)
s2 <- random_seq(slength = 10, swidth = 20,
    prob = c(0.6, 0.1, 0.3, 0))
s2

random_qual(  
  slength,  
  swidth,  
  qual = NULL,  
  encod = c("Sanger", "Illumina1.8", "Illumina1.5", "Illumina1.3", "Solexa"),  
  prob = NULL)

random_qual

Create random qualities for a given encoding

Description

Create a BStringSet object with random qualities

Usage

random_qual(  
  slength,  
  swidth,  
  qual = NULL,  
  encod = c("Sanger", "Illumina1.8", "Illumina1.5", "Illumina1.3", "Solexa"),  
  prob = NULL)
random_seq

Arguments

<table>
<thead>
<tr>
<th>Argument</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>slength</td>
<td>number of sequences</td>
</tr>
<tr>
<td>swidth</td>
<td>width of the sequences</td>
</tr>
<tr>
<td>qual</td>
<td>quality range for the sequences. It must be a range included in the selected encoding: 'Sanger' = [0, 40] 'Illumina1.8' = [0, 41] 'Illumina1.5' = [0, 40] 'Illumina1.3' = [3, 40] 'Solexa' = [-5, 40] example: for a range from 20 to 30 in Sanger encoding, pass the argument = c(20, 30)</td>
</tr>
<tr>
<td>encod</td>
<td>sequence encoding</td>
</tr>
<tr>
<td>prob</td>
<td>a vector of range = range(qual), with probabilities to set the frequency of each quality value. Default is equiprobability. If the sum of the probabilities is &gt; 1, the values will be normalized to the range [0, 1].</td>
</tr>
</tbody>
</table>

Value

BStringSet object

Author(s)

Leandro Roser <learoser@gmail.com>

Examples

```r
q <- random_qual(30, 20)
q
```

Description

Create a DNAStringSet object with random sequences

Usage

```r
random_seq(
  slength, swidth, nuc = c("DNA", "RNA"),
  prob = c(0.25, 0.25, 0.25, 0.25)
)
```
Arguments

- `slength`: Number of sequences
- `swidth`: Width of the sequences
- `nuc`: Create sequences of DNA (nucleotides = c('A', 'C', 'G', 'T')) or RNA (nucleotides = c('A', 'C', 'G', 'U'))?. Default: 'DNA'
- `prob`: A vector of four probability values used to set the frequency of the nucleotides 'A', 'C', 'G', 'T', for DNA, or 'A', 'C', 'G', 'U', for RNA. For example = c(0.25, 0.25, 0.5, 0). Default is = c(0.25, 0.25, 0.25, 0.25) (equiprobability for the 4 bases). If the sum of the probabilities is > 1, the values will be normalized to the range [0, 1].

Value

- DNAStringSet object

Author(s)

Leandro Roser <learoser@gmail.com>

Examples

```r
# For reproducible examples, make a call to set.seed before
# running each random function

set.seed(10)
s1 <- random_seq(slength = 10, swidth = 20)
s1

set.seed(10)
s2 <- random_seq(slength = 10, swidth = 20,
prob = c(0.6, 0.1, 0.3, 0))
s2
```

seq_filter

Remove a set of sequences

Description

Removes a set of sequences

Usage

`seq_filter(input, rm.seq)`
Arguments

input ShortReadQ object
rm.seq Character vector with sequences to remove

Value

Filtered ShortReadQ object

Author(s)

Leandro Roser <learoser@gmail.com>

Examples

require(ShortReadQ)
set.seed(10)
input <- random_length(30, 3:7)
rm.seq = c('TGGTC', 'CGGT', 'GTTCT', 'ATA')

# verify that some sequences match
match_before <- unlist(lapply(rm.seq, function(x) grep(x, as.character(sread(input)))))

filtered <- seq_filter(input, rm.seq = rm.seq)

# verify that matching sequences were removed
match_after <- unlist(lapply(rm.seq, function(x) grep(x, as.character(sread(filtered)))))

seq_names

Create sequences names

Description

Create BStringSet object with names

Usage

seq_names(n, base_name = "s", sep = ",")

Arguments

n Number of reads
base_name Base name for strings
sep Character separating base names and the read number. Default: ","
Value

BStringSet object

Examples

```r
snames <- seq_names(10)
snames
snames2 <- seq_names(10, base_name = 's', sep = '.

trim3q_filter

Filter sequences with low quality in 3' tails

Description

The program removes from the 3' tails of the sequences a set of nucleotides showing a quality < a threshold value in a ShortReadQ object

Usage

```r
trim3q_filter(
  input,
  rm.3qual,
  q_format = NULL,
  check.encod = TRUE,
  remove_zero = TRUE
)
```

Arguments

<table>
<thead>
<tr>
<th>Argument</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>input</td>
<td>ShortReadQ object</td>
</tr>
<tr>
<td>rm.3qual</td>
<td>Quality threshold for 3' tails</td>
</tr>
<tr>
<td>q_format</td>
<td>Quality format used for the file, as returned by check_encoding</td>
</tr>
<tr>
<td>check.encod</td>
<td>Check the encoding of the sequence? This argument is incompatible with q_format. Default TRUE</td>
</tr>
<tr>
<td>remove_zero</td>
<td>Remove zero-length sequences?</td>
</tr>
</tbody>
</table>

Value

Filtered ShortReadQ object

Author(s)

Leandro Roser <learoser@gmail.com>
Examples

```r
require('Biostrings')
require('ShortRead')

# create 6 sequences of width 20
set.seed(10)
input <- random_seq(6, 20)

# create qualities of width 15 and paste to qualities
# of length 5 used for the tails.
# for two of the sequences, put low qualities in tails
set.seed(10)
my_qual <- random_qual(c(30,40), slength = 6, swidth = 15,
                        encod = 'Sanger')

set.seed(10)
tails <- random_qual(c(30,40), slength = 6, swidth = 5,
                     encod = 'Sanger')

set.seed(10)
tails[2:3] <- random_qual(c(3, 20), slength = 2,
                          swidth = 5,  encod = 'Sanger')
my_qual <- paste0(my_qual, tails)

input_q <- BStringSet(my_qual)

# create names
input_names <- seq_names(6)

# create ShortReadQ object
my_read <- ShortReadQ(sread = input,
                      quality = input_q, id = input_names)

# apply the filter
filtered <- trim3q_filter(my_read, rm.3qual = 28)

# look at the trimmed sequences
sread(filtered)
```

---

**unique_filter**

Remove duplicated sequences in a FASTQ file

**Description**

This program is a wrapper to occurrenceFilter. It removes the duplicated sequences of a FASTQ file.

**Usage**

`unique_filter(input)`
Arguments
input ShortReadQ object

Value
Filtered ShortReadQ object

Author(s)
Leandro Roser <learoser@gmail.com>

Examples

```r
require('Biostrings')
require('ShortRead')

set.seed(10)
s <- random_seq(10, 10)
s <- sample(s, 30, replace = TRUE)
q <- random_qual(30, 10)
n <- seq_names(30)

my_read <- ShortReadQ(sread = s, quality = q, id = n)

# check presence of duplicates
isUnique(as.character(sread(my_read)))

# apply the filter
filtered <- unique_filter(my_read)

isUnique(as.character(sread(filtered)))
```
Index

* internal
  asc2int, 5
  check_onclick, 6
  create_cleanfunction, 8
  create_uniform_width, 8
  cutRseq, 9
  int2asc, 13
  isNaturalNumber, 13
  messageFun, 15
  myPlot, 16
  outputClean, 17
  plotA, 18
  plotB, 18
  plotC, 18
  plotD, 19
  plotE, 19
  plotF, 19
  plotG, 20
  plotH, 20
  plotI, 20
  plotJ, 21
  plotObjects, 21
  processingFunction, 22

adapter_filter, 3
asc2int, 5
BStringSet, 25, 26, 28, 29
check_encoding, 5
check_onclick, 6
complex_filter, 6
create_cleanfunction, 8
create_uniform_width, 8
cutLseq (cutRseq), 9
cutRseq, 9
DNAString, 3, 4, 10
DNAStringSet, 4, 10, 26, 27
fixed_filter, 11
injection_letter_random, 12
int2asc, 13
isNatural, 13
launch_fqc, 14
length_filter, 14
messageFun, 15
myPlot, 16
n_filter, 16
nFilter, 16
occurrenceFilter, 30
outputClean, 17
plotA, 18
plotB, 18
plotC, 18
plotD, 19
plotE, 19
plotF, 19
plotG, 20
plotH, 20
plotI, 20
plotJ, 21
plotObjects, 21
processingFunction, 22
qmean_filter, 22
random_length, 23
random_qual, 25
random_seq, 26
runApp, 14
seq_filter, 27
seq_names, 28
ShortReadQ, 3, 4, 7, 11, 15, 16, 22, 23, 25, 28, 29, 31
trim3q_filter, 29

32
INDEX

trimLRPatterns, 3, 4, 9, 10

unique_filter, 30