Package 'Rfastp'

April 18, 2025

Type Package

Title An Ultra-Fast and All-in-One Fastq Preprocessor (Quality Control, Adapter, low quality and polyX trimming) and UMI Sequence Parsing).

Version 1.19.0

Description Rfastp is an R wrapper of fastp developed in c++. fastp performs quality control for fastq files. including low quality bases trimming, polyX trimming, adapter auto-detection and trimming, paired-end reads merging, UMI sequence/id handling. Rfastp can concatenate multiple files into one file (like shell command cat) and accept multiple files as input.

License GPL-3 + file LICENSE

Encoding UTF-8

LazyData true

RoxygenNote 7.1.1

biocViews QualityControl, Sequencing, Preprocessing, Software

SystemRequirements GNU make

LinkingTo Rcpp, Rhtslib, zlibbioc

Imports Rcpp, rjson, ggplot2, reshape2

Suggests BiocStyle, testthat, knitr, rmarkdown

VignetteBuilder knitr

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

git_branch devel

git_last_commit c65390d

git_last_commit_date 2025-04-15

Repository Bioconductor 3.22

Date/Publication 2025-04-17

Author Wei Wang [aut] (ORCID: <https://orcid.org/0000-0002-3216-7118>), Ji-Dung Luo [ctb] (ORCID: <https://orcid.org/0000-0003-0150-1440>), Thomas Carroll [cre, aut] (ORCID: <https://orcid.org/0000-0002-0073-1714>)

Maintainer Thomas Carroll <tc.infomatics@gmail.com>

Contents

`catfastq` *Concatenate Fastq Files.*

Description

concatenate multiple fastq files into a single file.

Usage

```
catfastq(output, inputFiles, append = FALSE, paired = FALSE, shuffled = FALSE)
```

Arguments

`output`	output file name [string]
`inputFiles`	a vector of input file names [vector]
`append`	a logical indicating append the files to a file already exists.
`paired`	a logical indicating split the input files into two halves. the first half merged into read1, the second half merged into read2.
`shuffled`	a logical indicating split the input file list into two halves. The R1/R2 files are inteleaved in the inputFiles vector.

Value

no returns.

Author(s)

Wei Wang

Examples

```
pe001_read1 <- system.file("extdata","splited_001_R1.fastq.gz",
     package="Rfastp")
pe002_read1 <- system.file("extdata","splited_002_R1.fastq.gz",
     package="Rfastp")
pe003_read1 <- system.file("extdata","splited_003_R1.fastq.gz",
     package="Rfastp")
pe004_read1 <- system.file("extdata","splited_004_R1.fastq.gz",
     package="Rfastp")

pe001_read2 <- system.file("extdata","splited_001_R2.fastq.gz",
     package="Rfastp")
pe002_read2 <- system.file("extdata","splited_002_R2.fastq.gz",
```

```
     package="Rfastp")
pe003_read2 <- system.file("extdata","splited_003_R2.fastq.gz",
     package="Rfastp")
pe004_read2 <- system.file("extdata","splited_004_R2.fastq.gz",
     package="Rfastp")

allR1 <- c(pe001_read1, pe002_read1, pe003_read1, pe004_read1)
allR2 <- c(pe001_read2, pe002_read2, pe003_read2, pe004_read2)

allreads <- c(allR1, allR2)
allreads_shuffled <- c(pe001_read1, pe001_read2, pe002_read1, pe002_read2,
               pe003_read1, pe003_read2, pe004_read1, pe004_read2)

outputPrefix <- tempfile(tmpdir = tempdir())
# a normal concatenation for single-end libraries.

catfastq(output = paste0(outputPrefix, "_R1.fastq.gz"), inputFiles = allR1)

# a normal concatenation for paired-end libraries.

catfastq(output = paste0(outputPrefix, "merged_paired"),
    inputFiles = allreads, paired=TRUE)

# Append to exist files (paired-end)

catfastq(output=paste0(outputPrefix,"append_paired"), inputFiles=allreads,
    append=TRUE, paired=TRUE)

# Input paired-end files are shuffled.

catfastq(output=paste0(outputPrefix,"_shuffled_paired"),
    inputFiles=allreads_shuffled, paired=TRUE, shuffled=TRUE)
```

`curvePlot` *Plot of Base Quality and GC Content.*

Description

generate a ggplot2 object of Base Quality/GC content before and after QC.

Usage

```
curvePlot(json, curves = "quality_curves")
```

Arguments

`json`	the output json of function rfastq. [json]
`curves`	plots for Base Quality("quality_curves") or GC content("content_curves"). default is "quality_curves"

Value

a ggplot2 object.

Author(s)

Wei Wang

Examples

```
outputPrefix <- tempfile(tmpdir = tempdir())
se_read1 <- system.file("extdata","Fox3_Std_small.fq.gz",package="Rfastp")
se_json_report <- rfastp(read1 = se_read1, outputFastq = outputPrefix,
   thread = 4)
# Base Quality plot is the default output:
p1 <- curvePlot(se_json_report)
p1

p2 <- curvePlot(se_json_report, curves = "content_curves")
```

---

qcSummary *Summary of Fastq Quality Control*

Description

generate a data frame of the Fastq QC summary.

Usage

```
qcSummary(json)
```

Arguments

json	the output json of function rfastq. [json]

Value

a data frame.

Author(s)

Wei Wang

Examples

```
outputPrefix <- tempfile(tmpdir = tempdir())
se_read1 <- system.file("extdata","Fox3_Std_small.fq.gz",package="Rfastp")
se_json_report <- rfastp(read1 = se_read1, outputFastq = outputPrefix,
   thread = 4)
df_summary <- qcSummary(se_json_report)
```

rfastp *R wrap of fastp*

Description

Quality control (Cut adapter, low quality trimming, polyX trimming, UMI handling, and etc.) of fastq files.

Usage

```
rfastp(
  read1,
  read2 = "",
  outputFastq,
  unpaired = "",
  failedOut = "",
  merge = FALSE,
  mergeOut = "",
  phred64 = FALSE,
  interleaved = FALSE,
  fixMGIid = FALSE,
  adapterTrimming = TRUE,
  adapterSequenceRead1 = "auto",
  adapterSequenceRead2 = "auto",
  adapterFasta = "",
  trimFrontRead1 = 0,
  trimTailRead1 = 0,
  trimFrontRead2 = 0,
  trimTailRead2 = 0,
  maxLengthRead1 = 0,
  maxLengthRead2 = 0,
  forceTrimPolyG = FALSE,
  disableTrimPolyG = FALSE,
  minLengthPolyG = 10,
  trimPolyX = FALSE,
  minLengthPolyX = 10,
  cutWindowSize = 4,
  cutLowQualTail = FALSE,
  cutSlideWindowRight = FALSE,
  cutLowQualFront = FALSE,
  cutMeanQual = 20,
  cutFrontWindowSize = 4,
  cutFrontMeanQual = 20,
  cutTailWindowSize = 4,
  cutTailMeanQual = 20,
  cutSlideWindowSize = 4,
  cutSlideWindowQual = 20,
  qualityFiltering = TRUE,
  qualityFilterPhred = 15,
  qualityFilterPercent = 40,
  maxNfilter = 5,
```

```
  averageQualFilter = 0,
  lengthFiltering = TRUE,
  minReadLength = 15,
  maxReadLength = 0,
  lowComplexityFiltering = FALSE,
  minComplexity = 30,
  index1Filter = "",
  index2Filter = "",
  maxIndexMismatch = 0,
  correctionOverlap = FALSE,
  minOverlapLength = 30,
  maxOverlapMismatch = 5,
  maxOverlapMismatchPercentage = 20,
  umi = FALSE,
  umiLoc = "",
  umiLength = 0,
  umiPrefix = "",
  umiSkipBaseLength = 0,
  umiNoConnection = FALSE,
  umiIgnoreSeqNameSpace = FALSE,
  overrepresentationAnalysis = FALSE,
  overrepresentationSampling = 20,
  splitOutput = 0,
  splitByLines = 0,
  thread = 2,
  verbose = TRUE
)
```

Arguments

read1	read1 input file name(s). [vector]
read2	read2 input file name(s). [vector]
outputFastq	string of /path/prefix for output fastq [string]
unpaired	for PE input, output file name for reads which the mate reads failed to pass the QC [string], default NULL, discard it. [string]
failedOut	file to store reads that cannot pass the filters default NULL, discard it. [string]
merge	for PE input, A logical(1) indicating whether merge each pair of reads into a single read if they are overlaped, unmerged reads will be write to ‘output‘ file. Default is FALSE. the ‘mergeOut‘ must be set if TRUE.
mergeOut	under ‘merge‘ mode, file to store the merged reads. [string]
phred64	A logical indicating whether the input is using phred64 scoring (it will be converted to phred33, so the output will still be . phred33)
interleaved	A logical indicating whether <read1> is an interleaved FASTQ which contains both read1 and read2. Default is FALSE.
fixMGIid	the MGI FASTQ ID format is not compatible with many BAM operation tools, enable this option to fix it. Default is FALSE
adapterTrimming	A logical indicating whether run adapter trimming. Default is ‘TRUE‘
adapterSequenceRead1	the adapter for read1. For SE data, if not specified, the adapter will be auto-detected. For PE data, this is used if R1/R2 are found not overlapped.

`adapterSequenceRead2`
: the adapter for read2 (PE data only). This is used if R1/R2 are found not overlapped. If not specified, it will be the same as <adapterSequenceRead1>

`adapterFasta`
: specify a FASTA file to trim both read1 and read2 (if PE) by all the sequences in this FASTA file.

`trimFrontRead1`
: trimming how many bases in front for read1, default is 0.

`trimTailRead1`
: trimming how many bases in tail for read1, default is 0'

`trimFrontRead2`
: trimming how many bases in front for read2. If it's not specified, it will follow read1's settings

`trimTailRead2`
: trimming how many bases in tail for read2. If it's not specified, it will follow read1's settings

`maxLengthRead1`
: if read1 is longer than maxLengthRead1, then trim read1 at its tail to make it as long as maxLengthRead1 Default 0 means no limitation.

`maxLengthRead2`
: if read2 is longer than maxLengthRead2, then trim read2 at its tail to make it as long as maxLengthRead2. Default 0 means no limitation. If it's not specified, it will follow read1's settings.

`forceTrimPolyG`
: A logical indicating force polyG tail trimming, trimming is only automatically enabled for Illumina NextSeq/NovaSeq . data.

`disableTrimPolyG`
: A logical indicating disable polyG tail trimming.

`minLengthPolyG`
: the minimum length to detect polyG in the read tail. 10 by default.

`trimPolyX`
: A logical indicating force polyX tail trimming.

`minLengthPolyX`
: the minimum length to detect polyX in the read tail. 10 by default.

`cutWindowSize`
: the window size option shared by cutLowQualFront, cutLowQualTail, or cutSlideWindowRight. Range: 1~1000, default: 4

`cutLowQualTail`
: A logical indiccating move a sliding window from tail (3') to front, drop the bases in the window if its mean quality < threshold, stop otherwise. Default is 'FALSE'

`cutSlideWindowRight`
: A logical indicating move a sliding window from front to tail, if meet one window with mean quality < threshold, drop the bases in the window and the right part, and then stop. Default is 'FALSE'

`cutLowQualFront`
: A logical indiccating move a sliding window from front (5') to tail, drop the bases in the window if its mean quality < threshold, stop otherwise. Default is 'FALSE'

`cutMeanQual`
: the mean quality requirement option shared by cutLowQualFront, cutLowQualTail or cutSlideWindowRight. Range: 1~36, default: 20

`cutFrontWindowSize`
: the window size option of cutLowQualFront, default to cutWindowSize if not specified. default: 4

`cutFrontMeanQual`
: the mean quality requirement option for cutLowQualFront, default to cutMeanQual if not specified. default: 20

`cutTailWindowSize`
: the window size option of cutLowQualTail, default to cutWindowSize if not specified. default: 4

`cutTailMeanQual`
: the mean quality requirement option for cutLowQualTail, default to cutMeanQual if not specified. default: 20

`cutSlideWindowSize`
: the window size option of cutSlideWindowRight, default to cutWindowSize if not specified. default: 4

`cutSlideWindowQual`
: the mean quality requirement option for cutSlideWindowRight, default to cutMeanQual if not specified. default: 20

`qualityFiltering`
: A logical indicating run quality filtering. Default is 'TRUE'.

`qualityFilterPhred`
: the minimum quality value that a base is qualified. Default 15 means phred quality >=Q15 is qualified.

`qualityFilterPercent`
: Maximum percents of bases are allowed to be unqualified (0~100). Default 40 means 40%

`maxNfilter`
: maximum number of N allowed in the sequence. read/pair is discarded if failed to pass this filter. Default is 5

`averageQualFilter`
: if one read's average quality score < 'averageQualFilter', then this read/pair is discarded. Default 0 means no requirement.

`lengthFiltering`
: A logical indicating whether run lenght filtering. Default: TRUE

`minReadLength`
: reads shorter than minReadLength will be discarded, default is 15.

`maxReadLength`
: reads longer than maxReadLength will be discarded, default 0 means no limitation.

`lowComplexityFiltering`
: A logical indicating whethere run low complexity filter. The complexity is defined as the percentage of base that is different from its next base (base[i] != base[i+1]). Default is 'FALSE'

`minComplexity`
: the threshold for low complexity filter (0~100). Default is 30, which means 30% complexity is required. (int [=30])

`index1Filter`
: specify a file contains a list of barcodes of index1 to be filtered out, one barcode per line.

`index2Filter`
: specify a file contains a list of barcodes of index2 to be filtered out, one barcode per line.

`maxIndexMismatch`
: the allowed difference of index barcode for index filtering, default 0 means completely identical.

`correctionOverlap`
: A logical indicating run base correction in overlapped regions (only for PE data), default is 'FALSE'

`minOverlapLength`
: the minimum length to detect overlapped region of PE reads. This will affect overlap analysis based PE merge, adapter trimming and correction. 30 by default.

`maxOverlapMismatch`
: the maximum number of mismatched bases to detect overlapped region of PE reads. This will affect overlap analysis based PE merge, adapter trimming and correction. 5 by default.

`maxOverlapMismatchPercentage`
: the maximum percentage of mismatched bases to detect overlapped region of PE reads. This will affect overlap analysis based PE merge, adapter trimming and correction. Default 20 means 20%

`umi`
: A logical indicating whethere preprocessing unique molecular identifier (UMI). Default: 'FALSE'

`umiLoc`
: specify the location of UMI, can be (index1/index2/read1/read2/per_index/per_read)

`umiLength`
: length of UMI if the UMI is in read1/read2.

`umiPrefix`
: an string indication the following string is UMI (i.e. prefix=UMI, UMI=AATTCG, final=UMIAATTCG). Only letters, numbers, and '#" allowed. No prefix by default.

`umiSkipBaseLength`
: if the UMI is in read1/read2, skip 'umiSkipBaseLength' bases following UMI, default is 0.

`umiNoConnection`
: an logical indicating remove "_" between the UMI prefix string and the UMI string. Default is FALSE.

`umiIgnoreSeqNameSpace`
: an logical indicating ignore the space in the sequence name. Default is FALSE, the umi tag will be inserted into the sequence name before the first SPACE.

`overrepresentationAnalysis`
: A logical indicating overrepresentation analysis. Default is 'FALSE'

`overrepresentationSampling`
: one in 'overrepresentationSampling' reads will be computed for overrepresentation analysis (1~10000), smaller is slower, default is 20.

`splitOutput`
: number of files to be splitted (2~999). a sequential number prefix will be added to output name. Default is 0 (no split)

`splitByLines`
: split output by limiting lines of each file(>=1000), a sequential number prefix will be added to output name (0001.out.fq, 0002.out.fq...), default is 0 (disabled).

`thread`
: owrker thread number, default is 2

`verbose`
: output verbose log information

Value

returns a json object of the report.

Author(s)

Thomas Carroll, Wei Wang

Examples

```
# preprare for the input and output files.
# if the output file exists, it will be OVERWRITEN.
```

```
se_read1 <- system.file("extdata","Fox3_Std_small.fq.gz",package="Rfastp")
pe_read1 <- system.file("extdata","reads1.fastq.gz",package="Rfastp")
pe_read2 <- system.file("extdata","reads2.fastq.gz",package="Rfastp")
outputPrefix <- tempfile(tmpdir = tempdir())


# a normal single-end file

se_json_report <- rfastp(read1 = se_read1,
    outputFastq=paste0(outputPrefix, "_se"), thread = 4)


# merge paired-end data by overlap:

pe_json_report <- rfastp(read1 = pe_read1, read2 = pe_read2, merge = TRUE,
    outputFastq = paste0(outputPrefix, '_unpaired'),
    mergeOut = paste0(outputPrefix, '_merged.fastq.gz'))


# a clipr example

clipr_json_report <- rfastp(read1 = se_read1,
   outputFastq = paste0(outputPrefix, '_clipr'),
   disableTrimPolyG = TRUE,
   cutLowQualFront = TRUE,
   cutFrontWindowSize = 29,
   cutFrontMeanQual = 20,
   cutLowQualTail = TRUE,
   cutTailWindowSize = 1,
   cutTailMeanQual = 5,
   minReadLength = 29,
   adapterSequenceRead1 = 'GTGTCAGTCACTTCCAGCGG'
)
```

`trimSummary` *Summary of Fastq adapter and low quality trimming*

Description

generate a data frame of the Fastq trim summary.

Usage

```
trimSummary(json)
```

Arguments

`json`	the output json of function rfastq. [json]

Value

a data frame.

Author(s)

Wei Wang

Examples

```
outputPrefix <- tempfile(tmpdir = tempdir())
se_read1 <- system.file("extdata","Fox3_Std_small.fq.gz",package="Rfastp")
se_json_report <- rfastp(read1 = se_read1, outputFastq = outputPrefix,
   thread = 4, adapterSequenceRead1 = 'GTGTCAGTCACTTCCAGCGG')
trim_summary <- trimSummary(se_json_report)
```

Index