Package ‘QApckg’

May 12, 2022

Type Package

Title Quality assessment for Miseq data derived from viral sequencing

Version 0.1.0

Author Alicia Aranda Fernandez

Maintainer Alicia Aranda <ali.afernandez99@gmail.com>

Description This package provides a set of functions for NGS data processing,
quality analysis, filtering and demultiplexing. These functions are designed
to be applied in consecutive order on Miseq raw data to obtain a set of
intersected haplotypes for each evaluated sample. With this consensus haplotypes
different kinds of computations can be made, i.e genotyping, variant calling and
quasispecies diversity.

License file LICENSE

Encoding UTF-8

LazyData true

RoxygenNote 7.1.2

biocViews

git_url <https://github.com/aliafdz/QApckg>

Imports methods,
ape,
stats

Depends R (>= 4.1),
ShortRead,
RColorBrewer,
Biostrings,
stringr,
foreach,
doParallel,
QSutils,
muscle

Suggests knitr,
rmarkdown,
testthat (>= 3.0.0)

VignetteBuilder knitr

Config/testthat/edition 3

R topics documented:

`ConsHaplotypes` *Generate consensus haplotypes*

Description

Computes the intersection of forward and reverse strand haplotypes and generates some report files.

Usage

```
ConsHaplotypes(trimfiles, pm.res, thr = 0.2, min.seq.len = 150)
```

Arguments

`trimfiles`	Vector including the paths of demultiplexed files by specific primer, with fna extension.
`pm.res`	The list returned by `demultiplexPrimer`, including `fileTable` and `poolTable` data frames.
`thr`	Threshold to filter haplotypes at minimum abundance before multiple alignment.
`min.seq.len`	Threshold to filter haplotypes at minimum length before intersection.

Details

This function is designed to be used after the execution of `demultiplexPrimer` function from the same package. After the generation of FASTA files containing forward and reverse strand reads for the evaluated samples, `ConsHaplotypes` executes multiple alignment with `muscle` and returns the consensus haplotypes using `IntersectStrandHpls`, that will be saved using the helper function `SaveHaplotypes`.

Value

The function returns a `data.frame` object containing the intersection results for each combination of patient and pool, including the initial number of reads, filtered out reads (for being below a given frequency threshold or unique to a single strand), overlapping frequency between both strands and the common reads (in percentage and nº of reads).

After execution, two FASTA files for each combination of sample and pool will be saved in a newly generated MACH folder; the first includes multiple alignment between forward and reverse strand haplotypes, and the second includes the forward and reverse strands intersected. Additionaly, some report files will be generated in the reports folder:

1. `MA.Intersects-SummRprt.txt`: Includes the sumary results by reads number after abundance filter and strand intersection.
2. `MA.Intersects.plots.pdf`: Includes different barplots for each sample representing the frequency of forward, reverse and intersected strand haplotypes.
3. `IntersectBarplots.pdf`: Includes different barplots for all combinations of patient and pool, representing the number of intersected and filtered out reads, the intersection yield and global yield.

Note

A new file named `muscle.log` containing `muscle` options will be generated and saved in a folder named "tmp".

Author(s)

Alicia Aranda

See Also

`muscle`, `IntersectStrandHpls`, `demultiplexPrimer`, `SaveHaplotypes`

Examples

```
splitDir <- "./splits"
# Save the file names with complete path
splitfiles <- list.files(splitDir,recursive=TRUE,full.names=TRUE,include.dirs=TRUE)
# Get data
samples <- read.table("./data/samples.csv", sep="\t", header=T,
                      colClasses="character",stringsAsFactors=F)
mids <- read.table("./data/mids.csv", sep="\t", header=T,
                   stringsAsFactors=F)
# Apply previous function from QA analysis
pm.res <- demultiplexPrimer(splitfiles,samples,primers)
# Save the files generated by previous function
trimDir <- "./trim"
trimfiles <- list.files(trimDir,recursive=TRUE,full.names=TRUE,include.dirs=TRUE)
# Define necessary parameters
min.seq.len <- 150
thr <- 0.2
int.res <- ConsHaplotypes(trimfiles, pm.res, thr, min.seq.len)
```

`demultiplexMID` *Split reads by MID sequence*

Description

Demultiplex reads by identifying MID sequences within windows of expected positions in the sequenced reads. MIDs are 10 base-length oligonucleotides that allow the identification of samples from different patients or origins.

It is important to note that MID sequences will be not trimmed from reads, they are only identified for associate them with each sample.

Usage

```
demultiplexMID(
  flashffiles,
  samples,
  mids,
  maxdif = 1,
  mid.start = 1,
  mid.end = 40
)
```

Arguments

`flashffiles`	Vector including the paths of FLASH filtered files, with fastq extension.
`samples`	Data frame with relevant information about the samples of the sequencing experiment, including `Patient.ID`,`MID`,`Primer.ID`,`Region`,`RefSeq.ID`, and `Pool.Nm` columns.
`mids`	Data frame with the association between MID identifiers and their sequences.
`maxdif`	Number of mismatches allowed between MID and read sequences.
`mid.start`	Expected start position for MID in sequence.
`mid.end`	Expected end position for MID in sequence.

Value

A list containing the following:

`nreads`	A table with the number of reads identified for each MID.
`by.pools`	A table with the coverage of reads demultiplexed by pool.

After execution, a FASTA file for each combination of MID and pool will be saved in a splits folder (that will be created in working directory), including its associated reads. Additionaly, two report files will be generated in a reports folder:

1. `SplidByMIDs.barplots.pdf`: Includes a first barplot representing `nreads` data values, and a second plot with the `by.pools` data values.
2. `SplidByMIDs.Rprt.txt`: Includes the same data tables returned by the function.

Author(s)

Alicia Aranda

See Also

FiltbyQ30

Examples

```
flashFiltDir <- "./flashFilt"
# Save the file names with complete path
flashffiles <- list.files(flashFiltDir,recursive=TRUE,full.names=TRUE,include.dirs=TRUE)
# Get data
samples <- read.table("./data/samples.csv", sep="\t", header=T,
                      colClasses="character",stringsAsFactors=F)
mids <- read.table("./data/mids.csv", sep="\t", header=T,
                   stringsAsFactors=F)
# Set parameters
maxdif <- 1
mid.start <- 1
mid.end <- 40
dem.res<-demultiplexMID(flashffiles,samples,mids,maxdif,mid.start,mid.end)
```

---

demultiplexPrimer *Trim specific primer sequences*

Description

Demultiplex reads by identifying primer sequences within windows of expected positions in the sequenced reads. It is important to note that MID and primer sequences will be trimmed from reads after the identification of primers, but amplicon length is not predetermined.

Usage

```
demultiplexPrimer(
  splitfiles,
  samples,
  primers,
  prmm = 3,
  min.len = 180,
  target.st = 1,
  target.end = 100
)
```

Arguments

`splitfiles`	Vector including the paths of demultiplexed files by MID, with fna extension.
`samples`	Data frame with relevant information about the samples of the sequencing experiment, including `Patient.ID`,`MID`,`Primer.ID`,`Region`,`RefSeq.ID`, and `Pool.Nm` columns.
`primers`	Data frame with information about the *primers* used in the experiment, including `Ampl.Nm`,`Region`,`Primer.FW`,`Primer.RV`,`FW.pos`,`RV.pos`,`FW.tpos`,`RV.tpos`,`Aa.ipos`, and `Aa.lpos` columns.
`prmm`	Number of mismatches allowed between the primers and read sequences.

`min.len`	Minimum length desired for haplotypes. Any sequence below this length will be discarded.
`target.st, target.end`	Initial and end positions between which primer sequences will be searched.

Details

After demultiplexing reads by MID with `demultiplexMID` function, primer sequences are identified in both strands. First, forward strands are recognized by searching FW primer sequence in 5' end and the reverse complement of RV primer sequence in 3' end. Then, reverse strands are recognized by searching RV primer sequence in 5' end and FW primer sequence in 3' end, obtaining the reverse complement of all reads identified as reverse strands. So, both strands are obtained in a way that facilitates their intersection.

Value

A list containing the following:

`fileTable`	A table with relevant data of each FASTA file generated in execution, including their associated strand, mean read length, total reads and total haplotypes obtained.
`poolTable`	A table with the number of total trimmed reads and the yield of the process by pool.

After execution, a FASTA file for each combination of strand, MID and pool will be saved in a newly created trim folder, including its associated reads. Additionaly, some report files will be generated in a reports folder:

1. `AmpliconLengthsRprt.txt`: Includes the amplicon lengths of both strands for each sample (with their corresponding MID identifier).
2. `AmpliconLengthsPlot.pdf`: Includes a barplot for each sample representing the amplicon lengths of both strands.
3. `SplitByPrimersOnFlash.txt`: Includes a table of reads identified by primer, total reads identified by patient and the yield by pool.
4. `SplitByPrimersOnFlash.pdf,SplitByPrimersOnFlash-hz.pdf`: Includes some plots representing primer matches by patient (in nº of reads) and the coverage of forward/reverse matches by pool.
5. `SplittedReadsFileTable.txt`: A file containing the same information as `fileTable`.

Author(s)

Alicia Aranda

See Also

`demultiplexMID`, `primermatch`

Examples

```
# Set parameters
prmm <- 3
min.len <- 180
# The expected window for primer sequences will depend on the presence of
# adapters, MID sequences and/or M13 primer.
target.st <- 1
target.end <- 100
splitDir <- "./splits"
# Save the file names with complete path
splitfiles <- list.files(splitDir,recursive=TRUE,full.names=TRUE,include.dirs=TRUE)
# Get data
samples <- read.table("./data/samples.csv", sep="\t", header=T,
                      colClasses="character",stringsAsFactors=F)
mids <- read.table("./data/mids.csv", sep="\t", header=T,
                   stringsAsFactors=F)
pm.res <- demultiplexPrimer(splitfiles,samples,primers,prmm,min.len,target.st,target.end)
```

`executeFLASH` *Run FLASH to extend R1 and R2 reads.*

Description

With R1 and R2 files generated by paired-end sequencing, the function executes FLASH program to obtain the number of extended and not-extended reads.

Usage

```
executeFLASH(R1, R2, flash, flash.opts, outfile = "./flash.fastq")
```

Arguments

`R1`	Path for R1 reads file.
`R2`	Path for R2 reads file.
`flash`	Folder path containing FLASH executable.
`flash.opts`	Character indicating FLASH options that will be part of the execution command.
`outfile`	File path for FLASH output. If it is not specified, the fastq file generated will be saved in current working directory.

Value

This function returns a matrix containing de number of reads extended and not extended by FLASH. Additionaly, a fastq file with extended reads will be saved to outfile path. Further FLASH output files will be saved in a new folder named "tmp".

Note

This function is defined for correct execution of `R1R2toFLASH` function from the same package, where all arguments are defined automatically.

Author(s)

Alicia Aranda

See Also

`R1R2toFLASH`

`FiltbyQ30` *Filter haplotypes by Q30*

Description

This function applies `FastqStreamer` over a fastq file and removes all reads that have a defined fraction of bases below Q30. The remaining reads will be saved in a new fastq file.

Usage

```
FiltbyQ30(max.pct = 0.05, flashfiles, flashres, ncores = 1)
```

Arguments

`max.pct`	The maximum percentage of bases below Q30 allowed in reads (by default,5%).
`flashfiles`	Vector including the paths of files that are going to be processed, with fastq extension.
`flashres`	Table of results obtained after the execution of `R1R2toFLASH` function.
`ncores`	Number of cores to use for parallelization with `mclapply.hack`.

Details

This function is designed to be applied after `R1R2toFLASH` function from the same package. If `flashres` is not specified but FLASH extension was previously done, the function will try to load the FLASH results table from the reports folder.

Value

A `data.frame` object containing FLASH and Filtering results.

After the execution, a fastq file with remaining reads for each pool will be saved in a new flashFilt folder (if it is not previously created). Additionaly, two report files will be generated in a reports folder:

1. `FiltQ30.barplot.pdf`: Includes a first Bar plot representing raw reads, extended reads by FLASH and filtered reads, and a second Bar plot with the yield by process for each pool.
2. `FiltQ30_report.txt`: Includes the same data returned by the function.

The results table obtained includes two new columns with respect to FLASH results table, named FiltQ30 (number of filtered reads) and Raw (total sequencing reads).

Author(s)

Alicia Aranda

See Also

`R1R2toFLASH`, `FastqStreamer`

Examples

```
runDir <- "./run"
runfiles <- list.files(runDir,recursive=TRUE,full.names=TRUE,include.dirs=TRUE)
flash <- "./FLASH/flash.exe"
flashDir <- "./flash"
flashfiles <- list.files(flashDir,recursive=TRUE,full.names=TRUE,include.dirs=TRUE)
flashres <- R1R2toFLASH(runfiles,flash)
filtres <- FiltbyQ30(max.pct=0.05,flashfiles,flashres)
```

`GblYield` *Compute global yield by step*

Description

Generates global yield reports for each evaluated pool from previous results.

Usage

```
GblYield(samples, filtres, pm.res, int.res)
```

Arguments

`samples`	Data frame with relevant information about the samples of the sequencing experiment, including `Patient.ID`,`MID`,`Primer.ID`,`Region`,`RefSeq.ID`, and `Pool.Nm` columns.
`filtres`	The data frame returned by `FiltbyQ30` function.
`pm.res`	The list returned by `demultiplexPrimer`, including `fileTable` and `poolTable` data frames.
`int.res`	The data frame returned by `ConsHaplotypes` function.

Value

After execution, two report files will be saved in the reports folder:

1. `GlobalYieldBarplots.pdf`: Includes some barplots representing the yield (in nº of reads and percentage) by each step of the quality assessment pipeline. These representation is done for all pools included in the analysis and also for global results.
2. `GlobalYield-SumRprt.txt`: Summary report including global yield by analysis step in number of reads, in percentage by step and percentage referred to raw reads.

Note

This function is designed to be applied at the end of the quality assessment analysis and requires the previous execution of `FiltbyQ30`, `demultiplexPrimer` and `ConsHaplotypes` and functions from the same package.

Author(s)

Alicia Aranda

See Also

FiltbyQ30, demultiplexPrimer, ConsHaplotypes

Examples

```
## Execute FLASH extension
runDir <- "./run"
runfiles <- list.files(runDir,recursive=TRUE,full.names=TRUE,include.dirs=TRUE)
flash <- "./FLASH/flash.exe"
flashres <- R1R2toFLASH(runfiles,flash)

## Execute Q30 filtering
flashDir <- "./flash"
flashfiles <- list.files(flashDir,recursive=TRUE,full.names=TRUE,include.dirs=TRUE)
filtres <- FiltbyQ30(max.pct=0.05,flashfiles,flashres)

## Execute demultiplexing by MID with default parameters
flashFiltDir <- "./flashFilt"
flashffiles <- list.files(flashFiltDir,recursive=TRUE,full.names=TRUE,include.dirs=TRUE)
# Get data
samples <- read.table("./data/samples.csv", sep="\t", header=T,
                      colClasses="character",stringsAsFactors=F)
mids <- read.table("./data/mids.csv", sep="\t", header=T,
                   stringsAsFactors=F)
dem.res<-demultiplexMID(flashffiles,samples,mids)

## Execute demultiplexing by primer
splitDir <- "./splits"
splitfiles <- list.files(splitDir,recursive=TRUE,full.names=TRUE,include.dirs=TRUE)
pm.res <- demultiplexPrimer(splitfiles,samples,primers)

## Obtain consensus haplotypes (default parameters)
trimDir <- "./trim"
trimfiles <- list.files(trimDir,recursive=TRUE,full.names=TRUE,include.dirs=TRUE)
int.res <- ConsHaplotypes(trimfiles, pm.res, thr, min.seq.len)

## Apply function
GblYield(samples, filtres, pm.res, int.res)
```

---

mclapply.hack *Execution of parallel:mclapply() on Windows machines*

Description

Mimics forking on Windows machines just like it is done with mclapply in Mac or Linux.

Usage

```
mclapply.hack(..., mc.cores = NULL)
```

Arguments

mc.cores	Number of cores to use for parallelization.

Note

The function code was extracted from `post-10-mclapply-hack.R` written by Nathan VanHoudnos, and was only added in this package for parallelization using Windows operating system. See the original code in <https://github.com/nathanvan/mcmc-in-irt/blob/master/post-10-mclapply-hack.R>

Author(s)

Nathan VanHoudnos

See Also

`mclapply`

`PlotInDels` *Check for insertions and deletions in samples*

Description

Generates insertion/deletion plots for each evaluated sample from consensus haplotypes' sequences.

Usage

```
PlotInDels(machfiles, pm.res)
```

Arguments

`machfiles`	Vector including the paths of files generated by `ConsHaplotypes` function, with fna extension.
`pm.res`	The list returned by `demultiplexPrimer`, including `fileTable` and `poolTable` data frames.

Value

After execution, a file named `GapsBarPlots.pdf` will be saved in the reports folder, including the plots generated for all samples. In each plot, insertions are represented by red lines and deletions are represented by blue lines.

Note

This function is designed to be applied at the end of the quality assessment analysis and requires the previous execution of `demultiplexPrimer` and `ConsHaplotypes` functions from the same package.

Examples

```
## Execute demultiplexing by primer
splitDir <- "./splits"
splitfiles <- list.files(splitDir,recursive=TRUE,full.names=TRUE,include.dirs=TRUE)
pm.res <- demultiplexPrimer(splitfiles,samples,primers)

## Obtain consensus haplotypes (default parameters)
trimDir <- "./trim"
```

```
trimfiles <- list.files(trimDir,recursive=TRUE,full.names=TRUE,include.dirs=TRUE)
int.res <- ConsHaplotypes(trimfiles, pm.res, thr, min.seq.len)

## Apply function
mach.Dir <- "./MACH"
machfiles <- list.files(mach.Dir,recursive=TRUE, full.names=TRUE, include.dirs=TRUE)
PlotInDels(machfiles,pm.res)
```

PoolQCbyPos *Evaluate QC by position*

Description

This function evaluates fastq files before and after the execution of FLASH program to extend paired-end reads, and returns QC by position plots in pdf format.

It can be applied also after filtering FLASH fastq files by Phred Score.

Usage

```
PoolQCbyPos(flashfiles, samples, primers, runfiles, ncores = 1)
```

Arguments

`flashfiles`	Vector including the paths of FLASH processed/filtered files, with fastq extension.
`samples`	Data frame with relevant information about the samples of the sequencing experiment, including `Patient.ID,MID,Primer.ID,Region,RefSeq.ID`, and `Pool.Nm` columns.
`primers`	Data frame with information about the *primers* used in the experiment, including `Ampl.Nm,Region,Primer.FW,Primer.RV,FW.pos,RV.pos,FW.tpos,RV.tpos,Aa.ipos`, and `Aa.lpos` columns.
`runfiles`	Vector including the paths of Illumina MiSeq Raw Data files, often with fastq.gz extension. If the function is applied for filtered fastq files, this argument must be NA or missing.
`ncores`	Number of cores to use for parallelization with `mclapply.hack`.

Value

After execution, a pdf file for each pool used in the experiment will be saved in a reports folder (if it is not previously defined, the function will create this folder), and a message indicating that the files are generated will appear in console.

Author(s)

Alicia Aranda

Examples

```
runDir <- "./run"
flashDir <- "./flash"
repDir <- "./reports"
# Save the file names with complete path
runfiles <- list.files(runDir,recursive=TRUE,full.names=TRUE,include.dirs=TRUE)
flashfiles <- list.files(flashDir,recursive=TRUE,full.names=TRUE,include.dirs=TRUE)
# Get data
samples <- read.table("./data/samples.csv", sep="\t", header=T,
                     colClasses="character",stringsAsFactors=F)
primers <- read.table("./data/primers.csv", sep="\t", header=T,
                      stringsAsFactors=F)
PoolQCbyPos(flashfiles,samples,primers,runfiles)
```

`PoolQCbyRead` *Evaluate QC by read*

Description

This function evaluates fastq files after the execution of FLASH program to extend paired-end reads, and returns QC by read plots in pdf format.

Usage

```
PoolQCbyRead(flashfiles, samples, primers, ncores = 1)
```

Arguments

`flashfiles`	Vector including the paths of FLASH processed files, with fastq extension.
`samples`	Data frame with relevant information about the samples of the sequencing experiment, including `Patient.ID`,`MID`,`Primer.ID`,`Region`,`RefSeq.ID`, and `Pool.Nm` columns.
`primers`	Data frame with information about the *primers* used in the experiment, including `Ampl.Nm`,`Region`,`Primer.FW`,`Primer.RV`,`FW.pos`,`RV.pos`,`FW.tpos`,`RV.tpos`,`Aa.ipos`, and `Aa.lpos` columns.
`ncores`	Number of cores to use for parallelization with `mclapply.hack`.

Value

After execution, 3 pdf files will be saved in the reports folder; the first 2 include the QC by read plots for each pool used in the experiment, and the 3rd shows the read length distribution for each pool. A message indicating that the files are generated will appear in console.

Author(s)

Alicia Aranda

See Also

`R1R2toFLASH`, `QCscores`

Examples

```
flashDir <- "./flash"
repDir <- "./reports"
# Save the file names with complete path
flashfiles <- list.files(flashDir,recursive=TRUE,full.names=TRUE,include.dirs=TRUE)
# Get data
samples <- read.table("./data/samples.csv", sep="\t", header=T,
                     colClasses="character",stringsAsFactors=F)
primers <- read.table("./data/primers.csv", sep="\t", header=T,
                      stringsAsFactors=F)
PoolQCbyRead(flashfiles,samples,primers)
```

---

primermatch *Identfication of primer specific sequences*

Description

Identifies primer sequences in reads from a single sample and adds primer match results in defined tables.

Usage

```
primermatch(j, idx, flnms, pool)
```

Arguments

`j`	Integer corresponding to the sample (element in `idx`) to be evaluated.
`idx`	Vector with index of samples to be evaluated (for one pool).
`flnms`	Vector including the names of demultiplexed files by MID, with fna extension, corresponding to the evaluated samples.
`pool`	Character indicating the name of sample pool.

Details

This function is only defined for correct execution of `demultiplexPrimer` function from the same package, so it cannot be executed individually.

Value

This function requires the result tables named `FlTbl,PoolTbl` and `pr.res` that will be filled with the data collected from the evaluated sample. This results will be further evaluated in `demultiplexPrimer` parent function.

Author(s)

Alicia Aranda

See Also

`demultiplexPrimer`

QCplot	*QC plot by read position*

Description

Draws a scatter plot with Phred score values by read position.

Usage

```
QCplot(fvnm1, fvnm2, snm, SW = FALSE, FL = FALSE)
```

Arguments

`fvnm1, fvnm2`	Matrix or array containing Phred score values by read position for 0.05, 0.25, 0.5, 0.75 and 0.95 quantiles. If a QC by position plot for R1 and R2 files is required, both arguments are needed, one for each file.
`snm`	Character indicating the name of the pool of evaluated reads. Only required when both `fvnm1` and `fvnm2` are provided.
`SW`	Logical indicating if the plot should include quality profile by SW (Sliding Window). If `TRUE`, Phred Scores are computed as moving averages for windows of 10 base length slid along the sequence.
`FL`	Logical indicating if the first argument corresponds to the scores of FLASH extended reads. If `TRUE`, only `fvnm1` argument is required.

Details

`fvnm1` and `fvnm2` arguments are obtanied from the `QCscores` function with argument `byPos=TRUE`.

Value

A scatter plot with desired values will be shown in the active plots window.

Note

This function is only defined for correct execution of `PoolQCbyPos` function from the same package.

Author(s)

Alicia Aranda

See Also

`PoolQCbyPos`, `QCscores`

Examples

```
flashDir <- "./flash"
flashfiles <- list.files(flashDir,recursive=TRUE,full.names=TRUE,include.dirs=TRUE)
lst1 <- QCscores(file.path(flashDir,flashfiles[1]),byPos=T)
fvnm1 <- lst1$fvnq
QCplot(fvnm1,FL=TRUE) # QC plot for FLASH fastq file
QCplot(fvnm1,SW=TRUE,FL=TRUE) # QC plot by SW for FLASH fastq file
```

QCscores — *Compute Phred scores by position or by read*

Description

Applies FastqStreamer over a fastq file and returns quality control mesures (Phred scores), either by position or by read.

Usage

```
QCscores(flnm, ln = 301, byPos = FALSE, byRead = FALSE)
```

Arguments

flnm	Fastq file to be evaluated.
ln	Amplicon length.
byPos	Logical indicating if QC by position should be computed.
byRead	Logical indicating if QC by read should be computed. Note that Arguments byPos and byRead are mutually exclusive.

Value

If argument byPos=TRUE, the function returns a list including the following parameters:

1. fvnq: A matrix with Phred quality scores across each nucleotide base in the reads. Columns indicate base position and rows indicate 0.05, 0.25, 0.5, 0.75 and 0.95 Phred quantiles.
2. fvnl: A vector with normalized read lengths for each Phred quantile.
3. all.ln: A vector with all read lengths.

If argument byRead=TRUE, the function returns a list including the following parameters:

1. all.ln: A vector with all read lengths.
2. all.ln30: A vector with the number of bases below Q30 for each read.
3. all.fnl30: The result of dividing all.ln30/all.ln, which is the fraction of bases below Q30 for each read.

Note

This function is only defined for correct execution of PoolQCbyPos and PoolQCbyRead functions from the same package.

Author(s)

Alicia Aranda

See Also

PoolQCbyPos, PoolQCbyRead, QCplot

R1R2toFLASH — *Run FLASH to extend paired-end reads and generate report graphs.*

Description

This function applies `executeFLASH` over R1 and R2 reads for multiple sample pools and returns a file with the extended reads for each pool.

Usage

```
R1R2toFLASH(runfiles, flash, min.ov = 20, max.ov = 300, err.lv = 0.1)
```

Arguments

`runfiles`	Vector including the paths of Illumina MiSeq Raw Data files, often with fastq.gz extension.
`flash`	File path of FLASH executable.
`min.ov`	Minimum overlap (in nt) between R1 and R2 reads.
`max.ov`	Maximum overlap (in nt) between R1 and R2 reads.
`err.lv`	Mismatch fraction accepted in overlapping.

Value

The function returns a `data.frame` object containing FLASH results for sequenced regions.

After the execution, a fastq file with extended reads for each pool will be saved in a new folder named flash. Additionaly, two files will be saved in a reports folder:

1. `FLASH_barplot.pdf`: Bar plots representing extended vs not extended reads and the yield of the process for each pool.
2. `FLASH_report.txt`: Includes the data returned by the function with used FLASH parameters.

Author(s)

Alicia Aranda

See Also

`executeFLASH`

Examples

```
runDir <- "./run"
flash <- "./FLASH/flash.exe"
# Save the file names with complete path
runfiles<-list.files(runDir,recursive=TRUE,full.names=TRUE,include.dirs=TRUE)
min.ov <- 20
max.ov <- 300
err.lv <- 0.1
flashres <- R1R2toFLASH(runfiles,flash,min.ov,max.ov,err.lv)
```

SaveHaplotypes — *Save consensus haplotypes after sorting by mutation and abundance*

Description

Sorts and renames haplotypes by the number of mutations with respect to the dominant haplotype, and by abundance, and saves their sequences in a FASTA file.

Usage

```
SaveHaplotypes(flnm = "./SavedHaplotypes.fna", bseqs, nr, max.difs = 250)
```

Arguments

`flnm`	File name of the FASTA file that will be generated with haplotype sequences.
`bseqs`	Character object with the haplotype alignment.
`nr`	Vector with the haplotype counts.
`max.difs`	Maximum number of mismatches allowed with respect to the dominant haplotype.

Details

This function is similar to `SortByMutations` function from `QSutils` package but has new features. For example, in this case haplotypes with a huge number of mutations with respect to the dominant one are discarted, and columns with all gaps are eliminated. Also, the final sequences are saved in a FASTA file.

Value

A list containing the following:

`bseqs`	DNAStringSet or AAStringSet with the haplotype sequences.
`nr`	Vector of the haplotype counts.
`nm`	Vector of the number of differences of each haplotype with respect to the dominant haplotype.

After execution, a FASTA file named as `flnm` with the `bseqs` element will be generated.

Author(s)

Alicia Aranda

See Also

`SortByMutations`, `ConsHaplotypes`, `IntersectStrandHpls`

Index