Package 'polyester'

April 15, 2020

Maintainer Jack Fu <jmfu@jhsph.edu>, Jeff Leek <jtleek@gmail.com>

Author Alyssa C. Frazee, Andrew E. Jaffe, Rory Kirchner, Jeffrey T. Leek

Version 1.22.0

License Artistic-2.0

Title Simulate RNA-seq reads

Description This package can be used to simulate RNA-seq reads from differential expression experiments with replicates. The reads can then be aligned and used to perform comparisons of methods for differential expression.

VignetteBuilder knitr

Depends R (>= 3.0.0)

Imports Biostrings (>= 2.32.0), IRanges, S4Vectors, logspline, limma, zlibbioc

Suggests knitr, ballgown

LazyLoad true

biocViews Sequencing, DifferentialExpression

RoxygenNote 5.0.1

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

git_branch RELEASE_3_10

git_last_commit 3c4f995

git_last_commit_date 2019-10-29

Date/Publication 2020-04-14

R topics documented:

`add_error` *add sequencing error to simulated reads*

Description

simulate sequencing error by randomly changing the sequenced nucleotide on some of the reads

Usage

```
add_error(tFrags, error_rate = 0.005)
```

Arguments

`tFrags`	DNAStringSet representing sequencing reads
`error_rate`	error probability

Value

DNAStringSet equivalent to `tFrags` but with random sequencing errors inserted

Examples

```
library(Biostrings)
data(srPhiX174)
set.seed(174)
srPhiX174_withError = add_error(srPhiX174)
#error was introduced in, e.g., position 10 of 2nd string in set.
```

---

add_gc_bias *add GC bias to a count matrix*

Description

Given a matrix with rows corresponding to transcripts and sample-specific GC bias models, bias the count matrix using the bias model.

Usage

```
add_gc_bias(readmat, gcbias, transcripts)
```

Arguments

readmat	matrix of counts, with rows corresponding to features (transcripts) and columns corresponding to replicates
gcbias	List of GC bias models to add to readmat. Must have length equal to the number of columns of readmat. List elements must either be integers 0 through 7, where 0 means no bias and 1-7 correspond to built-in GC bias models, or objects of class loess which can predict a deviation from overall mean count (on the log scale) given a GC percentage between 0 and 1.
transcripts	DNAStringSet object containing the sequences of the features (transcripts) corresponding to the rows of readmat. Length must be equal to the number of rows in readmat.

Details

Designed for internal use in simulate_experiment functions.

Value

matrix of the same size as readmat, but with counts for each replicate biased according to gcbias.

Examples

```
library(Biostrings)
fastapath = system.file("extdata", "chr22.fa", package="polyester")
numtx = count_transcripts(fastapath)
transcripts = readDNAStringSet(fastapath)

# create a count matrix:
readmat = matrix(20, ncol=10, nrow=numtx)
readmat[1:30, 1:5] = 40

# add biases randomly: use built-in bias models
```

```
set.seed(137)
biases = sample(0:7, 10, replace=TRUE)
readmat_biased = add_gc_bias(readmat, as.list(biases), transcripts)
```

`add_platform_error` *Simulate sequencing error using empirical error model*

Description

Given a sequencing platform and a set of sequencing reads, add sequencing errors to the reads given a known error profile from the platform.

Usage

```
add_platform_error(tFrags, platform, paired, path = NULL)
```

Arguments

`tFrags`	DNAStringSetList containing error-free sequencing reads. If simulating a paired-end experiment, mate-pairs should appear next to each other in tFrags.
`platform`	Which sequencing platform should the error model be estimated from? Currently supports `'illumina4'`, `'illumina5'`, `'roche454'`, and `'custom'`.
`paired`	Does `tFrags` contain paired end reads, with mate pairs next to each other? (TRUE if yes.)
`path`	if platform is `'custom'`, provide the path to the error model. After processing the error model with `build_error_models.py`, you will have either two files (ending in _mate1 and _mate2, if your model was for paired-end reads) or one file (ending in _single, if your model was for single-end reads). The `path` argument should be the path to the error model *up to but not including* _mate1/_mate2/_single.

Details

This function adds sequencing error to a set of reads based on the position in the read and the true nucleotide at that location. Position-specific probabilities of making each possible sequencing error (reading a T when it should have been A, reading a G when it should have been T, etc.) were calculated for each of three platforms using the empirical error models available with the GemSIM software (see references). Users can also estimate an error model from their own data using GemSIM and can use that error model with Polyester as described in the vignette. (You will need to run a Python script available at the Polyester GitHub repository to process the error model).

Value

`DNAStringSet` object that is the same as `tFrags` except but with sequencing error added.

References

McElroy KE, Luciani F and Thomas T (2012): GemSIM: general, error-model based simulator of next-generation sequencing data. BMC Genomics 13(1), 74.

See Also

add_error for uniform error

Examples

```
library(Biostrings)
# pretend the srPhiX174 DNAStringSet represents 35bp single-end
# sequencing reads:
data(srPhiX174)
set.seed(718)
data_with_errors = add_platform_error(srPhiX174, 'illumina4', paired=FALSE)

# the 17th read in this set has an error at position 20:
data_with_errors[17][[1]][20] # N
srPhiX174[17][[1]][20] # T

# 101 reads total have at least one sequencing error:
sum(data_with_errors != srPhiX174)
```

---

cdnaf *Model of positional bias that can arise when RNA-seq is performed using protocols relying on cDNA fragmentation.*

Description

This positional bias model was estimated in Li and Jiang (2012). With cDNA fragmentation, reads are more likely to have come from the 3' end of the transcript. The probabilities included in this dataset were estimated from Supplementary Figure S3 in Li and Jiang's manuscript. Data points from the figure were inferred and exported as CSV files using WebPlotDigitizer. The CSV files and the code used to process them and create the datasets are available in the Polyester GitHub repository (https://github.com/alyssafrazee/polyester).

Format

data frame with 100 rows and 2 columns. Column 1 is position along a transcript (in percent), while Column 2 is the probability of getting a fragment at that position. Column 2 sums to 1.

References

Li W and Jiang T (2012): Transcriptome assembly and isoform expression level estimation from biased RNA-Seq reads. Bioinformatics 28(22): 2914-2921.

Rohatgi A (2014): WebPlotDigitizer: Version 3.4 of WebPlotDigitizer. ZENODO. 10.5281/zenodo.11835

count_transcripts *determine how many transcripts are annotated in a FASTA or GTF file*

Description

determine how many transcripts are annotated in a FASTA or GTF file

Usage

```
count_transcripts(f, fasta = TRUE, identifier = "transcript_id",
  attrsep = "; ")
```

Arguments

f	character, path to a file in FASTA or GTF format
fasta	TRUE if f is a fasta file; FALSE if f is a GTF file
identifier	if f is a GTF file, how are transcripts identified in the attributes field (9th column) of the file? Default transcript_id.
attrsep	if f is a GTF file, how are attributes separated in the attributes field (9th column) of the file? Default "; ".

Value

Number of transcripts annotated in f

Examples

```
fastapath = system.file("extdata", "chr22.fa", package="polyester")
count_transcripts(fastapath) #918
```

create_read_numbers *Generate a simulated data set based on known model parameters*

Description

Generate a simulated data set based on known model parameters

Usage

```
create_read_numbers(mu, fit, p0, m = NULL, n = NULL, mod = NULL,
  beta = NULL, seed = NULL)
```

Arguments

mu	Baseline mean expression for negative binomial model
fit	Fitted relationship between log mean and log size
p0	A vector of the probabilities a count is zero
m	Number of genes/transcripts to simulate (not necessary if mod, beta are specified)
n	Number of samples to simulate (not necessary if mod, beta are specified)
mod	Model matrix you would like to simulate from without an intercept
beta	set of coefficients for the model matrix (must have same number of columns as mod)
seed	optional seed to set (for reproducibility)

Value

counts Data matrix with counts for genes in rows and samples in columns

Author(s)

Jeff Leek

Examples

```
library(ballgown)
data(bg)
countmat = fpkm_to_counts(bg, mean_rps=400000)
params = get_params(countmat)
Ntranscripts = 50
Nsamples = 10
custom_readmat = create_read_numbers(mu=params$mu, fit=params$fit,
  p0=params$p0, m=Ntranscripts, n=Nsamples, seed=103)
```

---

empirical_density *Estimated distribution of fragment lengths*

Description

Empirical fragment length distribution was estimated using 7 randomly selected RNA-seq samples from the GEUVADIS dataset ('t Hoen et al 2013). One sample was selected from each of the 7 laboratories that performed the sequencing. We used Picard's "CollectInsertSizeMetrics" tool (http://broadinstitude.github.io/picard/), version 1.121, to estimate the fragment size distribution based on read alignments. Code we used to estimate this distribution is available at https://github.com/alyssafrazee/polyester/blob/master/make_fraglen_model.R.

Format

logspline object (created with logspline) specifying the empirical density of fragment lengths in the 7 GEUVADIS samples.

References

't Hoen PA, et al (2013): Reproducibility of high-throughput mRNA and small RNA sequencing across laboratories. Nature Biotechnology 31(11): 1015-1022.

`fpkm_to_counts` *Turn FPKMs from a ballgown object into estimated counts for transcripts*

Description

Turn FPKMs from a ballgown object into estimated counts for transcripts

Usage

```
fpkm_to_counts(bg = NULL, mat = NULL, tlengths = NULL, mean_rps = 1e+08,
  threshold = 0)
```

Arguments

`bg`	ballgown object created from real RNA-seq dataset
`mat`	matrix of isoform-level FPKMs from which to derive counts. Rows should represent transcripts and columns should represent counts. Provide exactly one of `bg` or `mat`.
`tlengths`	if using `mat` instead of `bg`, vector of transcript lengths. Entries correspond to the rows of `mat`. Lengths should only count the nucleotides within transcripts' exons.
`mean_rps`	This should be the number of reads per sample in total for use in backing out the FPKM calculations.
`threshold`	only estimate parameters from transcripts with mean FPKM measurements at least as large as `threshold`.

Details

If transcripts/exons are represented by `GRanges` or `GRangesList` objects, the `width` function is really useful in calculating transcript lengths.

Value

A matrix of counts with the same number of rows and columns as the ballgown object

Author(s)

Jeff Leek

Examples

```
library(ballgown)
data(bg)
countmat = fpkm_to_counts(bg, mean_rps=400000)
```

`generate_fragments` *generate a set of fragments from a set of transcripts*

Description

Convert each sequence in a DNAStringSet to a "fragment" (subsequence)

Usage

```
generate_fragments(tObj, fraglen = 250, fragsd = 25, readlen = 100,
  distr = "normal", custdens = NULL, bias = "none",
  frag_GC_bias = "none")
```

Arguments

`tObj`	DNAStringSet of sequences from which fragments should be extracted
`fraglen`	Mean fragment length, if drawing fragment lengths from a normal distribution.
`fragsd`	Standard deviation of fragment lengths, if drawing lengths from a normal distribution. Note: `fraglen` and `fragsd` are ignored unless `distr` is 'normal'.
`readlen`	Read length. Default 100. Used only to label read positions.
`distr`	One of 'normal', 'empirical', or 'custom'. If 'normal', draw fragment lengths from a normal distribution with mean `fraglen` and standard deviation `fragsd`. If 'empirical', draw fragment lengths from a fragment length distribution estimated from a real data set. If 'custom', draw fragment lengths from a custom distribution, provided as the `custdens` argument, which should be a density fitted using `logspline`.
`custdens`	If `distr` is 'custom', draw fragments from this density. Should be an object of class `logspline`.
`bias`	One of 'none', 'rnaf', or 'cdnaf' (default 'none'). 'none' represents uniform fragment selection (every possible fragment in a transcript has equal probability of being in the experiment); 'rnaf' represents positional bias that arises in protocols using RNA fragmentation, and 'cdnaf' represents positional bias arising in protocols that use cDNA fragmentation (Li and Jiang 2012). Using the 'rnaf' model, coverage is higher in the middle of the transcript and lower at both ends, and in the 'cdnaf' model, coverage increases toward the 3' end of the transcript. The probability models used come from Supplementary Figure S3 of Li and Jiang (2012).
`frag_GC_bias`	See explanation in `simulate_experiment`.

Details

The empirical fragment length distribution was estimated using 7 randomly selected RNA-seq samples from the GEUVADIS dataset ('t Hoen et al 2013), one sample from each laboratory that performed sequencing for that data set. We used Picard's "CollectInsertSizeMetrics" (http://broadinstitute.github.io/picard/), version 1.121, to estimate the insert size distribution based on the read alignments.

Value

DNAStringSet consisting of one randomly selected subsequence per element of `tObj`.

References

't Hoen PA, et al (2013): Reproducibility of high-throughput mRNA and small RNA sequencing across laboratories. Nature Biotechnology 31(11): 1015-1022.

Li W and Jiang T (2012): Transcriptome assembly and isoform expression level estimation from biased RNA-Seq reads. Bioinformatics 28(22): 2914-2921.

See Also

logspline

Examples

```
library(Biostrings)
data(srPhiX174)

## get fragments with lengths drawn from normal distrubution
set.seed(174)
srPhiX174_fragments = generate_fragments(srPhiX174, fraglen=15, fragsd=3,
    readlen=4)
srPhiX174_fragments
srPhiX174

## get fragments with lengths drawn from an empirical distribution
empirical_frags = generate_fragments(srPhiX174, distr='empirical')
empirical_frags

## get fragments with lengths from a normal distribution, but include
## positional bias from cDNA fragmentation:
biased_frags = generate_fragments(srPhiX174, bias='cdnaf')
biased_frags
```

`getAttributeField` *extract a specific field of the "attributes" column of a data frame created from a GTF/GFF file*

Description

extract a specific field of the "attributes" column of a data frame created from a GTF/GFF file

Usage

```
getAttributeField(x, field, attrsep = "; ")
```

Arguments

`x`	vector representing the "attributes" column of GTF/GFF file
`field`	name of the field you want to extract from the "attributes" column
`attrsep`	separator for the fields in the attributes column. Defaults to '; ', the separator for GTF files outputted by Cufflinks.

Value

vector of nucleotide positions included in the transcript

Author(s)

Wolfgang Huber, in the davidTiling package (LGPL license)

See Also

http://useast.ensembl.org/info/website/upload/gff.html, for specifics of the GFF/GTF file format.

Examples

```
library(ballgown)
gtfPath = system.file('extdata', 'annot.gtf.gz', package='ballgown')
gffdata = gffRead(gtfPath)
gffdata$transcriptID = getAttributeField(gffdata$attributes,
  field = "transcript_id")
```

---

get_params — *Estimate zero-inflated negative binomial parameters from a real dataset*

Description

This function estimates the parameters of a zero inflated negative binomial distribution based on a real count data set based on the method of moments. The function also returns a spline fit of log mean to log size which can be used when generating new simulated data.

Usage

```
get_params(counts, threshold = NULL)
```

Arguments

counts	A matrix of counts. If you want to simulate from a ballgown object, see fpkm_to_counts
threshold	Only estimate parameters from transcripts with row means greater than threshold

Value

p0 A vector of probabilities that the count will be zero, one for each gene/transcript.

mu The estimated negative binomial mean by method of moments for the non-zero counts

size The estimated negative binomial size by method of moments for the non-zero counts

fit A fit relating log mean to log size for use in simulating new data.

Author(s)

Jeff Leek

Examples

```
library(ballgown)
data(bg)
countmat = fpkm_to_counts(bg, mean_rps=400000)
params = get_params(countmat)
```

`get_reads` *get sequencing reads from fragments*

Description

simulate the sequencing process by returning the sequence of one or both ends of provided fragments

Usage

```
get_reads(tFrags, readlen, paired = TRUE)
```

Arguments

`tFrags`	DNAStringSet representing fragments
`readlen`	Read length.
`paired`	If `FALSE`, return only the first `readlen` bases of each element of `tFrags` in the result; if `TRUE`, also return last `readlen` bases.

Value

DNAStringSet representing simulated RNA-seq reads

See Also

`simulate_experiment`, `simulate_experiment_countmat`

Examples

```
library(Biostrings)
data(srPhiX174)
set.seed(174)
srPhiX174_reads = get_reads(srPhiX174, readlen=15, paired=FALSE)
srPhiX174_reads
# set of single-end, 15bp reads, treating srPhiX174 as the fragments
```

gtf_dataframe *data frame (in gtf-inspired format) for chromosome 22, hg19*

Description

In the data frame `gtf_dataframe`, each row corresponds to an exon / coding sequence / start codon / stop codon, and the columns correspond to standard GTF columns denoting annotated genomic features. See http://www.ensembl.org/info/website/upload/gff.html.

Format

data frame, 9 columns, 17769 rows

Source

Illumina iGenomes, hg19, 6 March 2013 version: http://ccb.jhu.edu/software/tophat/igenomes.shtml.

loessfit1 *Empirical GC bias model, NA06985*

Description

Loess model for log counts measuring transcript expression as a function of the transcript's GC content. The model was created using sample NA06985 in the `Ballgown` obtained at http://files.figshare.com/1625419/fpkm.rda

Format

Object of class `loess`

Source

Constructed using the code available at https://github.com/alyssafrazee/polyester/blob/master/gc_bias.R

References

GEUVADIS data set: 't Hoen PA, et al (2013): Reproducibility of high-throughput mRNA and small RNA sequencing across laboratories. Nature Biotechnology 31(11): 1015-1022.

Lappalainen, et al (2013): Transcriptome and genome sequencing uncovers functional variation in humans. Nature 501: 506-511.

loessfit2 *Empirical GC bias model, NA12144*

Description

Loess model for log counts measuring transcript expression as a function of the transcript's GC content. The model was created using sample NA12144 in the Ballgown obtained at http://files.figshare.com/1625419/fpkm.rda

Format

Object of class loess

Source

Constructed using the code available at https://github.com/alyssafrazee/polyester/blob/master/gc_bias.R

References

GEUVADIS data set: 't Hoen PA, et al (2013): Reproducibility of high-throughput mRNA and small RNA sequencing across laboratories. Nature Biotechnology 31(11): 1015-1022.

Lappalainen, et al (2013): Transcriptome and genome sequencing uncovers functional variation in humans. Nature 501: 506-511.

loessfit3 *Empirical GC bias model, NA12776*

Description

Loess model for log counts measuring transcript expression as a function of the transcript's GC content. The model was created using sample NA12776 in the Ballgown obtained at http://files.figshare.com/1625419/fpkm.rda

Format

Object of class loess

Source

Constructed using the code available at https://github.com/alyssafrazee/polyester/blob/master/gc_bias.R

References

GEUVADIS data set: 't Hoen PA, et al (2013): Reproducibility of high-throughput mRNA and small RNA sequencing across laboratories. Nature Biotechnology 31(11): 1015-1022.

Lappalainen, et al (2013): Transcriptome and genome sequencing uncovers functional variation in humans. Nature 501: 506-511.

loessfit4 *Empirical GC bias model, NA18858*

Description

Loess model for log counts measuring transcript expression as a function of the transcript's GC content. The model was created using sample NA18858 in the Ballgown obtained at http://files.figshare.com/1625419/fpkm.rda

Format

Object of class loess

Source

Constructed using the code available at https://github.com/alyssafrazee/polyester/blob/master/gc_bias.R

References

GEUVADIS data set: 't Hoen PA, et al (2013): Reproducibility of high-throughput mRNA and small RNA sequencing across laboratories. Nature Biotechnology 31(11): 1015-1022.

Lappalainen, et al (2013): Transcriptome and genome sequencing uncovers functional variation in humans. Nature 501: 506-511.

loessfit5 *Empirical GC bias model, NA20542*

Description

Loess model for log counts measuring transcript expression as a function of the transcript's GC content. The model was created using sample NA20542 in the Ballgown obtained at http://files.figshare.com/1625419/fpkm.rda

Format

Object of class loess

Source

Constructed using the code available at https://github.com/alyssafrazee/polyester/blob/master/gc_bias.R

References

GEUVADIS data set: 't Hoen PA, et al (2013): Reproducibility of high-throughput mRNA and small RNA sequencing across laboratories. Nature Biotechnology 31(11): 1015-1022.

Lappalainen, et al (2013): Transcriptome and genome sequencing uncovers functional variation in humans. Nature 501: 506-511.

loessfit6 *Empirical GC bias model, NA20772*

Description

Loess model for log counts measuring transcript expression as a function of the transcript's GC content. The model was created using sample NA20772 in the Ballgown obtained at http://files.figshare.com/1625419/fpkm.rda

Format

Object of class loess

Source

Constructed using the code available at https://github.com/alyssafrazee/polyester/blob/master/gc_bias.R

References

GEUVADIS data set: 't Hoen PA, et al (2013): Reproducibility of high-throughput mRNA and small RNA sequencing across laboratories. Nature Biotechnology 31(11): 1015-1022.

Lappalainen, et al (2013): Transcriptome and genome sequencing uncovers functional variation in humans. Nature 501: 506-511.

loessfit7 *Empirical GC bias model, NA20815*

Description

Loess model for log counts measuring transcript expression as a function of the transcript's GC content. The model was created using sample NA20815 in the Ballgown obtained at http://files.figshare.com/1625419/fpkm.rda

Format

Object of class loess

Source

Constructed using the code available at https://github.com/alyssafrazee/polyester/blob/master/gc_bias.R

References

GEUVADIS data set: 't Hoen PA, et al (2013): Reproducibility of high-throughput mRNA and small RNA sequencing across laboratories. Nature Biotechnology 31(11): 1015-1022.

Lappalainen, et al (2013): Transcriptome and genome sequencing uncovers functional variation in humans. Nature 501: 506-511.

`model1` *Empirical error model for Illumina Genome Analyzer IIx with Illumina Sequencing Kit v4 chemistry, read mate 1 of a pair*

Description

for each position in mate 1 of a paired-end read generated with the specified Illumina chemistry, this data frame contains the probability of not making a sequencing error, and of making each of the 4 possible types of sequencing errors. The reference base (truth) is in column 1, and the probabilities of sequencing that base given its read position (column 7) as each of the 5 possible bases (A, T, G, C, and N) is given in columns 2 through 6, respectively. So for example, at position 8 in mate 1 of a read where the true base is A, the probability of correctly calling that base an A is 0.9998, the probability of making an error by sequencing a T is 2.64e-05, the probability of making an error by sequencing a G is 1.58e-04, the probability of making an error by sequencing a C is 3.05e-05, and the probability of reading an 'N' at position 8 is 0. This can be seen by looking at `model1[model1$pos == 8,]`. Note that position indexing is 1-based, though a 0 position is included as described in the GemSIM documentation.

Format

data frame named `model1`, 7 columns, 505 rows

Source

processed from the Illumina v4 error model that ships with GemSIM (see references)

References

McElroy KE, Luciani F, Thomas T (2012). GemSIM: general, error-model based simulator of next-generation sequencing data. BMC Genomics 13(1), 74.

`model2` *Empirical error model for Illumina Genome Analyzer IIx with Illumina Sequencing Kit v4 chemistry, read mate 2 of a pair*

Description

for each position in mate 2 of a paired-end read generated with the specified Illumina chemistry, this data frame contains the probability of not making a sequencing error, and of making each of the 4 possible types of sequencing errors. The reference base (truth) is in column 1, and the probabilities of sequencing that base given its read position (column 7) as each of the 5 possible bases (A, T, G, C, and N) is given in columns 2 through 6, respectively. So for example, at position 8 in mate 1 of a read where the true base is A, the probability of correctly calling that base an A is 0.9995, the probability of making an error by sequencing a T is 0.00017, the probability of making an error by sequencing a G is 0.00023, the probability of making an error by sequencing a C is 6.02e-05, and the probability of reading an 'N' at position 8 is 1.15e-05. This can be seen by looking at `model2[model2$pos == 8,]`. Note that position indexing is 1-based, though a 0 position is included as described in the GemSIM documentation.

Format

data frame named `model2`, 7 columns, 505 rows

Source

processed from the Illumina v4 error model that ships with GemSIM (see references)

References

McElroy KE, Luciani F, Thomas T (2012). GemSIM: general, error-model based simulator of next-generation sequencing data. BMC Genomics 13(1), 74.

`model3` *Empirical error model for Illumina Genome Analyzer IIx with Illumina Sequencing Kit v4 chemistry, single-end read*

Description

for each position in a single-end read generated with the specified Illumina chemistry, this data frame contains the probability of not making a sequencing error, and of making each of the 4 possible types of sequencing errors. The reference base (truth) is in column 1, and the probabilities of sequencing that base given its read position (column 7) as each of the 5 possible bases (A, T, G, C, and N) is given in columns 2 through 6, respectively. So for example, at position 8 in mate 1 of a read where the true base is A, the probability of correctly calling that base an A is 0.9998, the probability of making an error by sequencing a T is 2.95e-05, the probability of making an error by sequencing a G is 1.27e-04, the probability of making an error by sequencing a C is 1.85e-05, and the probability of reading an 'N' at position 8 is 0. This can be seen by looking at `model3[model3$pos == 8,]`. Note that position indexing is 1-based, though a 0 position is included as described in the GemSIM documentation.

Format

data frame named `model3`, 7 columns, 505 rows

Source

processed from the Illumina v4 error model that ships with GemSIM (see references)

References

McElroy KE, Luciani F, Thomas T (2012). GemSIM: general, error-model based simulator of next-generation sequencing data. BMC Genomics 13(1), 74.

`model4` *Empirical error model for Illumina Genome Analyzer IIx with TrueSeq SBS Kit v5-GA chemistry, read mate 1 of a pair*

Description

for each position in mate 1 of a paired-end read generated with the specified Illumina chemistry, this data frame contains the probability of not making a sequencing error, and of making each of the 4 possible types of sequencing errors. The reference base (truth) is in column 1, and the probabilities of sequencing that base given its read position (column 7) as each of the 5 possible bases (A, T, G, C, and N) is given in columns 2 through 6, respectively. So for example, at position 8 in mate 1 of a read where the true base is A, the probability of correctly calling that base an A is 0.9998, the probability of making an error by sequencing a T is 4.00e-05, the probability of making an error by sequencing a G is 1.58e-04, the probability of making an error by sequencing a C is 1.46e-05, and the probability of reading an 'N' at position 8 is 0. This can be seen by looking at `model4[model4$pos == 8,]`. Note that position indexing is 1-based, though a 0 position is included as described in the GemSIM documentation.

Format

data frame named `model4`, 7 columns, 505 rows

Source

processed from the Illumina v5 error model that ships with GemSIM (see references)

References

McElroy KE, Luciani F, Thomas T (2012). GemSIM: general, error-model based simulator of next-generation sequencing data. BMC Genomics 13(1), 74.

`model5` *Empirical error model for Illumina Genome Analyzer IIx with TrueSeq SBS Kit v5-GA chemistry, read mate 2 of a pair*

Description

for each position in mate 2 of a paired-end read generated with the specified Illumina chemistry, this data frame contains the probability of not making a sequencing error, and of making each of the 4 possible types of sequencing errors. The reference base (truth) is in column 1, and the probabilities of sequencing that base given its read position (column 7) as each of the 5 possible bases (A, T, G, C, and N) is given in columns 2 through 6, respectively. So for example, at position 8 in mate 1 of a read where the true base is A, the probability of correctly calling that base an A is 0.9992, the probability of making an error by sequencing a T is 0.0002, the probability of making an error by sequencing a G is 0.0002, the probability of making an error by sequencing a C is 0.0001, and the probability of reading an 'N' at position 8 is 0.0002. This can be seen by looking at `model5[model5$pos == 8,]`. Note that position indexing is 1-based, though a 0 position is included as described in the GemSIM documentation.

Format

data frame named `model5`, 7 columns, 505 rows

Source

processed from the Illumina v5 error model that ships with GemSIM (see references)

References

McElroy KE, Luciani F, Thomas T (2012). GemSIM: general, error-model based simulator of next-generation sequencing data. BMC Genomics 13(1), 74.

`model6` *Empirical error model for Illumina Genome Analyzer IIx with TrueSeq SBS Kit v5-GA chemistry, single-end read*

Description

for each position in a single-end read generated with the specified Illumina chemistry, this data frame contains the probability of not making a sequencing error, and of making each of the 4 possible types of sequencing errors. The reference base (truth) is in column 1, and the probabilities of sequencing that base given its read position (column 7) as each of the 5 possible bases (A, T, G, C, and N) is given in columns 2 through 6, respectively. So for example, at position 8 in mate 1 of a read where the true base is A, the probability of correctly calling that base an A is 0.9998, the probability of making an error by sequencing a T is 3.04e-05, the probability of making an error by sequencing a G is 1.36e-04, the probability of making an error by sequencing a C is 1.27e-05, and the probability of reading an 'N' at position 8 is 0. This can be seen by looking at `model6[model6$pos == 8,]`. Note that position indexing is 1-based, though a 0 position is included as described in the GemSIM documentation.

Format

data frame named `model6`, 7 columns, 505 rows

Source

processed from the Illumina v5 error model that ships with GemSIM (see references)

References

McElroy KE, Luciani F, Thomas T (2012). GemSIM: general, error-model based simulator of next-generation sequencing data. BMC Genomics 13(1), 74.

model7 *Empirical error model Roche/454 FLX Titanium, single-end read*

Description

for each position in a single-end read generated with the specified chemistry, this data frame contains the probability of not making a sequencing error, and of making each of the 4 possible types of sequencing errors. The reference base (truth) is in column 1, and the probabilities of sequencing that base given its read position (column 7) as each of the 5 possible bases (A, T, G, C, and N) is given in columns 2 through 6, respectively. So for example, at position 8 in mate 1 of a read where the true base is C, the probability of correctly calling that base a C is 0.9994, the probability of making an error by sequencing a T is 0.0002, the probability of making an error by sequencing a G is 0.0001, the probability of making an error by sequencing an A is 0.0002, and the probability of reading an 'N' at position 8 is 0. This can be seen by looking at `model7[model7$pos == 8,]`. Note that position indexing is 1-based, though a 0 position is included as described in the GemSIM documentation.

Format

data frame named `model7`, 7 columns, 505 rows

Source

processed from the Roche 454 error model that ships with GemSIM (see references)

References

McElroy KE, Luciani F, Thomas T (2012). GemSIM: general, error-model based simulator of next-generation sequencing data. BMC Genomics 13(1), 74.

NB *Draw nonzero negative binomial random numbers*

Description

Draw nonzero negative binomial random numbers

Usage

```
NB(basemeans, size, seed = NULL)
```

Arguments

`basemeans`	vector of means, one per draw
`size`	vector of size parameters (controlling the mean/variance relationship); one per draw
`seed`	optional seed to set before drawing

Value

vector of negative binomial draws from specified distributions, where any zero draw is replaced with a 1. Length of return vector is equal to `length(basemeans)`.

Examples

```
randomNBs = NB(c(100, 4, 29), size=c(50, 2, 4), seed=21)
randomNBs  # 115, 5, 15
```

---

polyester *Polyester: simulating RNA-seq reads including differential expression*

Description

Polyester is an R package designed to simulate an RNA sequencing experiment. Given a set of annotated transcripts, polyester will simulate the steps of an RNA-seq experiment (fragmentation, reverse-complementing, and sequencing) and produce files containing simulated RNA-seq reads. Simulated reads can be analyzed using any of several downstream analysis tools.

Details

A single function call produces RNA-seq reads in FASTA format from a case/control experiment including biological replicates. Differential expression between cases and controls can be set by the user, facilitating comparisons of statistical differential expression methods for RNA-seq data. See detailed documentation for `simulate_experiment` and `simulate_experiment_countmat`.

See the vignette by typing `browseVignettes("polyester")` in the R prompt.

Author(s)

Alyssa Frazee, Andrew Jaffe, Rory Kirchner, Jeff Leek

References

Alyssa C Frazee, Geo Pertea, Andrew E Jaffe, Ben Langmead, Steven L Salzberg, Jeffrey T Leek (2014). Flexible isoform-level differential expression analysis with Ballgown. BioRxiv preprint: http://biorxiv.org/content/early/2014/03/30/003665.

---

reverse_complement *reverse-complement some fragments*

Description

randomly reverse-complement half of the sequences in a DNAStringSet

Usage

```
reverse_complement(tObj, seed = NULL)
```

Arguments

tObj	DNAStringSet representing sequences.
seed	optional seed to set before randomly selecting the sequences to be reverse-complemented.

Value

DNAStringSet that is the same as tObj, but with about half the sequences reverse-complemented.

Examples

```
library(Biostrings)
data(srPhiX174)
srPhiX174_halfrc = reverse_complement(srPhiX174, seed=174)
```

---

rnaf *Model of positional bias that can arise when RNA-seq is performed using protocols relying on RNA fragmentation.*

Description

This positional bias model was estimated in Li and Jiang (2012). With RNA fragmentation, reads are more likely to have come from the middle of the transcript than either end. The probabilities included in this dataset were estimated from Supplementary Figure S3 in Li and Jiang's manuscript. Data points from the figure were inferred and exported as CSV files using WebPlotDigitizer. The CSV files and the code used to process them and create the datasets are available in the Polyester GitHub repository (https://github.com/alyssafrazee/polyester).

Format

data frame with 100 rows and 2 columns. Column 1 is position along a transcript (in percent), while Column 2 is the probability of getting a fragment at that position. Column 2 sums to 1.

References

Li W and Jiang T (2012): Transcriptome assembly and isoform expression level estimation from biased RNA-Seq reads. Bioinformatics 28(22): 2914-2921.

Rohatgi A (2014): WebPlotDigitizer: Version 3.4 of WebPlotDigitizer. ZENODO. 10.5281/zenodo.11835

seq_gtf *Get transcript sequences from GTF file and sequence info*

Description

Given a GTF file (for transcript structure) and DNA sequences, return a DNAStringSet of transcript sequences

Usage

```
seq_gtf(gtf, seqs, feature = "transcript", exononly = TRUE,
  idfield = "transcript_id", attrsep = "; ")
```

Arguments

gtf	one of path to GTF file, or data frame representing a canonical GTF file.
seqs	one of path to folder containing one FASTA file (`.fa` extension) for each chromosome in `gtf`, or named DNAStringSet containing one DNAString per chromosome in `gtf`, representing its sequence. In the latter case, `names(seqs)` should contain the same entries as the `seqnames` (first) column of `gtf`.
feature	one of `'transcript'` or `'exon'` (default transcript), depending on desired return.
exononly	if `TRUE` (as it is by default), only create transcript sequences from the features labeled `exon` in `gtf`.
idfield	in the `attributes` column of `gtf`, what is the name of the field identifying transcripts? Should be character. Default `"transcript_id"`.
attrsep	in the `attributes` column of `gtf`, how are attributes separated? Default `"; "`.

Value

If feature is `'transcript'`, DNAStringSet containing transcript sequences, with names corresponding to `idfield` in `gtf`. If feature is `'exon'`, DNAStringSet containing exon sequences from `gtf`, named by exon location (chr, start, end, strand).

References

http://www.ensembl.org/info/website/upload/gff.html

Examples

```
 ## Not run:
  library(Biostrings)
  system('wget https://www.dropbox.com/s/04i6msi9vu2snif/chr22seq.rda')
  load('chr22seq.rda')
  data(gtf_dataframe)
  chr22_processed = seq_gtf(gtf_dataframe, chr22seq)

## End(Not run)
```

simulate_experiment *simulate RNA-seq experiment using negative binomial model*

Description

create FASTA files containing RNA-seq reads simulated from provided transcripts, with optional differential expression between two groups

Usage

```
simulate_experiment(fasta = NULL, gtf = NULL, seqpath = NULL,
  outdir = ".", num_reps = c(10, 10), reads_per_transcript = 300,
  size = NULL, fold_changes, paired = TRUE, reportCoverage = FALSE, ...)
```

Arguments

fasta	path to FASTA file containing transcripts from which to simulate reads. See details.
gtf	path to GTF file containing transcript structures from which reads should be simulated. See details.
seqpath	path to folder containing one FASTA file (.fa extension) for each chromosome in gtf. See details.
outdir	character, path to folder where simulated reads should be written, with *no* slash at the end. By default, reads are written to current working directory.
num_reps	How many biological replicates should be in each group? The length num_reps determines how many groups are in the experiment. For example, num_reps = c(5,6,5) specifies a 3-group experiment with 5 samples in group 1, 6 samples in group 2, and 5 samples in group 3. Defaults to a 2-group experiment with 10 reps per group (i.e., c(10,10)).
reads_per_transcript	baseline mean number of reads to simulate from each transcript. Can be an integer, in which case this many reads are simulated from each transcript, or an integer vector whose length matches the number of transcripts in fasta. Default 300. You can also leave reads_per_transcript empty and set meanmodel=TRUE to draw baseline mean numbers from a model based on transcript length.
size	the negative binomial size parameter (see NegBinomial) for the number of reads drawn per transcript. It can be a matrix (where the user can specify the size parameter per transcript, per group), a vector (where the user can specify the size per transcript, perhaps relating to reads_per_transcript), or a single number, specifying the size for all transcripts and groups. If left NULL, defaults to reads_per_transcript * fold_changes / 3. Negative binomial variance is mean + mean^2 / size.
fold_changes	Matrix specifying multiplicative fold changes between groups. There is no default, so you must provide this argument. In real data sets, lowly-expressed transcripts often show high fold changes between groups, so this can be kept in mind when setting fold_changes and reads_per_transcript. This argument must have the same number of columns as there are groups as specified by num_reps, and must have the same number of rows as there are transcripts in fasta. A fold change of X in matrix entry i,j means that for replicate j, the

	baseline mean number of reads (reads_per_transcript[i]) will be multiplied by X. Note that the multiplication happens before the negative binomial value (for the number of reads that *actually will* be drawn from transcript i, for replicate j) is drawn. This argument is ignored if `length(num_reps)` is 1 (meaning you only have 1 group in your simulation).
`paired`	If `TRUE`, paired-end reads are simulated; else single-end reads are simulated. Default `TRUE`
`reportCoverage`	whether to write out coverage information to `sample_coverages.rda` file in the `outdir`. defaults to `FALSE`
`...`	any of several other arguments that can be used to add nuance to the simulation. See details.

Details

Reads can either be simulated from a FASTA file of transcripts (provided with the `fasta` argument) or from a GTF file plus DNA sequences (provided with the `gtf` and `seqpath` arguments). Simulating from a GTF file and DNA sequences may be a bit slower: it took about 6 minutes to parse the GTF/sequence files for chromosomes 1-22, X, and Y in hg19.

Several optional parameters can be passed to this function to adjust the simulation. The options are:

- `readlen`: read length. Default 100.
- `lib_sizes`: Library size factors for the biological replicates. `lib_sizes` should have length equal to the total number of replicates in the experiment, i.e., `sum(num_reps)`. For each replicate, once the number of reads to simulate from each transcript for that replicate is known, all read numbers across all transcripts from that replicate are multiplied by the corresponding entry in `lib_sizes`.
- `distr` One of 'normal', 'empirical', or 'custom', which specifies the distribution from which to draw RNA fragment lengths. If 'normal', draw fragment lengths from a normal distribution. You can provide the mean of that normal distribution with `fraglen` (defaults to 250) and the standard deviation of that normal distribution with `fragsd` (defaults to 25). You can provide a single number for each, or a vector with length equal to the total number of samples. If 'empirical', draw fragment lengths from a fragment length distribution estimated from a real data set. If 'custom', draw fragment lengths from a custom distribution, which you can provide as the `custdens` argument. `custdens` should be a density fitted using `logspline`.
- `error_model`: The error model can be one of:
  - `'uniform'`: errors are distributed uniformly across reads. You can also provide an `'error_rate'` parameter, giving the overall probability of making a sequencing error at any given nucleotide. This error rate defaults to 0.005.
  - `'illumina4'` or `'illumina5'`: Empirical error models. See `?add_platform_error` for more information.
  - `'custom'`: A custom error model you've estimated from an RNA-seq data set using `GemErr`. See `?add_platform_error` for more info. You will need to provide both `model_path` and `model_prefix` if using a custom error model. `model_path` is the output folder you provided to `build_error_model.py`. This path should contain either two files suffixed _mate1 and _mate2, or a file suffixed _single. `model_prefix` is the 'prefix' argument you provided to `build_error_model.py` and is whatever comes before the _mate1/_mate2 or _single files in `model_path`.
- `bias` One of 'none', 'rnaf', or 'cdnaf'. 'none' represents uniform fragment selection (every possible fragment in a transcript has equal probability of being in the experiment); 'rnaf'

represents positional bias that arises in protocols using RNA fragmentation, and 'cdnaf' represents positional bias arising in protocols that use cDNA fragmentation (Li and Jiang 2012). Using the 'rnaf' model, coverage is higher in the middle of the transcript and lower at both ends, and in the 'cdnaf' model, coverage increases toward the 3' end of the transcript. The probability models used come from Supplementary Figure S3 of Li and Jiang (2012). Defaults to 'none' if you don't provide this.

- `gcbias` list indicating which samples to add GC bias to, and from which models. Should be the same length as `sum(num_reps)`; entries can be either numeric or of class `loess`. A numeric entry of 0 indicates no GC bias. Numeric entries 1 through 7 correspond to the 7 empirical GC models that ship with Polyester, estimated from GEUVADIS HapMap samples NA06985, NA12144, NA12776, NA18858, NA20542, NA20772, and NA20815, respectively. The code used to derive the empirical GC models is available at https://github.com/alyssafrazee/polyester/blob/master/make_gc_bias.R. A loess entry should be a loess prediction model that takes a GC content percent value (between 0 and 1) a transcript's deviation from overall mean read count based on that GC value. Counts for each replicate will be adjusted based on the GC bias model specified for it. Numeric and loess entries can be mixed. By default, no bias is included.
- `frag_GC_bias` Either a matrix of dimensions 101 x `sum(num_reps)` or 'none'. The default is 'none'. If specified, the matrix contains the probabilities (a number in the range [0,1]) that a fragment will appear in the output given its GC content. The first row corresponds to a fragment with GC content of 0 percent, the second row 1 percent, the third row 2 percent, etc., and the last row 100 percent. The columns correspond to different probabilites for each sample. Internally, a coin flip (a Bernoulli trial) determines if each fragment is kept, depending on its GC content. Note that the final library size will depend on the elements of the matrix, and it might make sense to scale up the `lib_size` of the samples with low probabilites in the matrix in the range of the transcriptome GC content distribution. Note that the `count_matrix` written to `outdir` contains the counts before applying fragment GC bias.
- `strand_specific` defaults to `FALSE`, which means fragments are generated with equal probability from both strands of the transcript sequence. set to `TRUE` for strand-specific simulation (1st read forward strand, 2nd read reverse strand with respect to transcript sequence).
- `meanmodel`: set to TRUE if you'd like to set `reads_per_transcripts` as a function of transcript length. We fit a linear model regressing transcript abundance on transcript length, and setting `meanmodel=TRUE` means we will use transcript lengths to draw transcript abundance based on that linear model. You can see our modeling code at http://htmlpreview.github.io/?https://github.com/alyssafrazee/polyester_code/blob/master/length_simulation.html
- `write_info`: set to FALSE if you do not want files of simulation information written to disk. By default, transcript fold changes and expression status, replicate library sizes and group identifiers, and an R data object of the counts matrix (before application of fragment GC bias) are written to `outdir`.
- `seed`: specify a seed (e.g. `seed=142` or some other integer) to set before randomly drawing read numbers, for reproducibility.
- `transcriptid`: optional vector of transcript IDs to be written into `sim_info.txt` and used as transcript identifiers in the output fasta files. Defaults to `names(readDNAStringSet(fasta))`. This option is useful if default names are very long or contain special characters.
- `gzip`: pass `gzip=TRUE` to write gzipped fasta files as output (by default, fasta output files are not compressed when written to disk).
- `exononly`: (passed to `seq_gtf`) if `TRUE` (as it is by default), only create transcript sequences from the features labeled `exon` in `gtf`.

- idfield: (passed to seq_gtf)in the attributes column of gtf, what is the name of the field identifying transcripts? Should be character. Default "transcript_id".
- attrsep: (passed to seq_gtf) in the attributes column of gtf, how are attributes separated? Default "; ".

Value

No return, but simulated reads and a simulation info file are written to outdir. Note that reads are written out transcript by transcript and so need to be shuffled if used as input to quantification algorithms such as eXpress or Salmon.

References

't Hoen PA, et al (2013): Reproducibility of high-throughput mRNA and small RNA sequencing across laboratories. Nature Biotechnology 31(11): 1015-1022.

Li W and Jiang T (2012): Transcriptome assembly and isoform expression level estimation from biased RNA-Seq reads. Bioinformatics 28(22): 2914-2921.

McElroy KE, Luciani F and Thomas T (2012): GemSIM: general, error-model based simulator of next-generation sequencing data. BMC Genomics 13(1), 74.

Examples

```
## simulate a few reads from chromosome 22

fastapath = system.file("extdata", "chr22.fa", package="polyester")
numtx = count_transcripts(fastapath)
set.seed(4)
fold_change_values = sample(c(0.5, 1, 2), size=2*numtx,
   prob=c(0.05, 0.9, 0.05), replace=TRUE)
fold_changes = matrix(fold_change_values, nrow=numtx)
library(Biostrings)
# remove quotes from transcript IDs:
tNames = gsub("'", "", names(readDNAStringSet(fastapath)))

simulate_experiment(fastapath, reads_per_transcript=10,
   fold_changes=fold_changes, outdir='simulated_reads',
   transcriptid=tNames, seed=12)
```

---

simulate_experiment_countmat

Simulate RNA-seq experiment

Description

create FASTA files containing RNA-seq reads simulated from provided transcripts, with optional differential expression between two groups (designated via read count matrix)

Usage

```
simulate_experiment_countmat(fasta = NULL, gtf = NULL, seqpath = NULL,
  readmat, outdir = ".", paired = TRUE, seed = NULL, ...)
```

Arguments

fasta	path to FASTA file containing transcripts from which to simulate reads. See details.
gtf	path to GTF file or data frame containing transcript structures from which reads should be simulated. See details and seq_gtf.
seqpath	path to folder containing one FASTA file (.fa extension) or DNAStringSet containing one entry for each chromosome in gtf. See details and seq_gtf.
readmat	matrix with rows representing transcripts and columns representing samples. Entry i,j specifies how many reads to simulate from transcript i for sample j.
outdir	character, path to folder where simulated reads should be written, without a slash at the end of the folder name. By default, reads written to the working directory.
paired	If TRUE, paired-end reads are simulated; else single-end reads are simulated.
seed	Optional seed to set before simulating reads, for reproducibility.
...	Additional arguments to add nuance to the simulation, as described extensively in the details of simulate_experiment, or to pass to seq_gtf, if gtf is not NULL.

Details

Reads can either be simulated from a FASTA file of transcripts (provided with the fasta argument) or from a GTF file plus DNA sequences (provided with the gtf and seqpath arguments). Simulating from a GTF file and DNA sequences may be a bit slower: it took about 6 minutes to parse the GTF/sequence files for chromosomes 1-22, X, and Y in hg19.

Value

No return, but simulated reads are written to outdir.

References

Li W and Jiang T (2012): Transcriptome assembly and isoform expression level estimation from biased RNA-Seq reads. Bioinformatics 28(22): 2914-2921.

Examples

```
fastapath = system.file("extdata", "chr22.fa", package="polyester")
numtx = count_transcripts(fastapath)
readmat = matrix(20, ncol=10, nrow=numtx)
readmat[1:30, 1:5] = 40

simulate_experiment_countmat(fasta=fastapath,
  readmat=readmat, outdir='simulated_reads_2', seed=5)
```

simulate_experiment_empirical *Simulate RNA-seq experiment based on abundances from a data set*

Description

Create fasta files representing reads from a two-group experiment, where abundances and differential expression are estimated from a real data set

Usage

```
simulate_experiment_empirical(bg = NULL, fpkmMat = NULL, mean_rps = 5e+06,
  grouplabels = NULL, decut = 1.5, outdir = ".", ...)
```

Arguments

bg	Ballgown object containing estimated transcript abundances in FPKM. Reads will be simulated for the same number of replicates that are in bg. Must provide exactly one of bg and fpkmMat.
fpkmMat	transcript-by-sample matrix containing abundances (in FPKM) estimated from a real data set. MUST have row names identifying transcripts. The number of columns is the number of samples that will be simulated.
mean_rps	Number of reads per sample to use in converting FPKM measurements to counts. Should be somewhat close to the number of reads per sample in the experiment that generated the estimated FPKMs. Defaults to 5 million (5e6).
grouplabels	vector indicating the group labels for each replicate in the experiment. Must be convertible to a factor with exactly 2 levels.
decut	A transcript will be recorded as truly differentially expressed if its fold change between the two groups is more extreme than decut, in either direction.
outdir	character, path to folder where simulated reads should be written, without a slash at the end of the folder name. By default, reads written to the working directory.
...	Additional arguments to pass to simulate_experiment_countmat

Value

No return, but reads are written to outdir.

Examples

```
## Not run:

  library(ballgown)
  data(bg)
  bg = subset(bg, "chr=='22'")

  # load gtf file:
  gtfpath = system.file('extdata', 'bg.gtf.gz', package='polyester')
  gtf = subset(gffRead(gtfpath), seqname=='22')

  # load/download chromosome sequence (just for this example)
```

```
  system('wget https://www.dropbox.com/s/04i6msi9vu2snif/chr22seq.rda')
  load('chr22seq.rda')
  names(chr22seq) = '22'

  # simulate reads based on this experiment's FPKMs
  simulate_experiment_empirical(bg, grouplabels=pData(bg)$group, gtf=gtf,
     seqpath=chr22seq, mean_rps=5000, outdir='simulated_reads_3', seed=1247)

## End(Not run)
```

`write_reads` *write sequencing reads to disk*

Description

given a DNAStringSet representing simulated sequencing reads, write FASTA files to disk representing the simulated reads.

Usage

```
write_reads(reads, fname, readlen, paired = TRUE, gzip, offset = 1L)
```

Arguments

`reads`	DNAStringSet representing sequencing reads
`fname`	file path/prefix specifying where sequencing reads should be written. Should not contain ".fasta" (this is appended automatically).
`readlen`	maximum length of the reads in `reads`.
`paired`	If `TRUE`, reads are assumed to be in pairs: i.e., read 1 and read 2 in `reads` are the left and right mate (respectively) of a read pair; same with read 3 and read 4, etc. The odd-numbered reads are written to `fname_1.fasta` and the even-numbered reads are written to `fname_2.fasta`. If `FALSE`, reads are assumed to be single-end and just one file, `fname.fasta`, is written.
`gzip`	If `TRUE`, gzip the output fasta files.
`offset`	An integer number greater or equal to 1 to start assigning read numbers at.

Details

The `get_reads` function returns a DNAStringSet object representing sequencing reads that can be directly passed to `write_reads`. If output other than that from `get_reads` is used and `paired` is `TRUE`, make sure `reads` is ordered properly (i.e., that mate pairs appear together and that the left mate appears first).

Value

No return, but FASTA file(s) containing the sequences in `reads` are written to `fname.fasta` (if `paired` is FALSE) or `fname_1.fasta` and `fname_2.fasta` if `paired` is TRUE.

See Also

`get_reads`

Examples

```
  library(Biostrings)
  data(srPhiX174) # pretend srPhiX174 represents a DNAStringSet of *reads*
  readlen = unique(width(srPhiX174)) #35
  write_reads(srPhiX174, fname='./srPhiX174', readlen=readlen, paired=FALSE,
      gzip=FALSE)

## If the file is too big, you can subset it and write it in chunks.
## Here we split our 'reads' into two chunks and save them to the same file.
write_reads(srPhiX174[1:100], fname='./srPhiX174-offset', readlen=readlen,
   paired=FALSE, gzip=FALSE, offset = 1L)
write_reads(srPhiX174[101:length(srPhiX174)], fname='./srPhiX174-offset',
   readlen=readlen, paired=FALSE, gzip=FALSE, offset = 101L)

## We can verify that we get the same results
srPhi <- readDNAStringSet('./srPhiX174.fasta')
srPhiOffset <- readDNAStringSet('./srPhiX174-offset.fasta')
identical(srPhi, srPhiOffset)
```

Index