Package 'polyester'

October 12, 2016

Maintainer Alyssa Frazee <`alyssa.frazee@gmail.com`>, Jeff Leek <`jtleek@gmail.com`>

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

Version 1.8.3

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 BiocGenerics, Biostrings (>= 2.32.0), IRanges, S4Vectors, logspline, limma

Suggests knitr, ballgown

biocViews Sequencing, DifferentialExpression

NeedsCompilation no

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

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

`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)
```

`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, mean_rps = 1e+08, threshold = 0)
```

Arguments

`bg`	ballgown object created from real RNA-seq dataset
`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 larger than `threshold`

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, fragsd = 25)
```

Arguments

`tObj`	DNAStringSet of sequences from which fragments should be extracted
`fraglen`	Mean fragment length.
`fragsd`	Standard deviation of fragment length. Fragment lengths are drawn from a normal distribution with mean `fraglen` and standard deviation `fragsd`.

Value

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

Examples

```
library(Biostrings)
  data(srPhiX174)
  set.seed(174)
  srPhiX174_fragments = generate_fragments(srPhiX174, fraglen=15, fragsd=3)
  srPhiX174_fragments
  srPhiX174
```

`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.

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

```
## Not run:
  randomNBs = NB(c(100, 4, 29), size=c(50, 2, 4), seed=21)
  randomNBs  # 115, 5, 15

## End(Not run)
```

`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)

Andrew Jaffe, Alyssa Frazee, 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)
```

`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, 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`.
`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

DNAStringSet containing transcript sequences, with names corresponding to `idfield` in `gtf`

References

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

Examples

```
library(Biostrings)
  load(url('http://biostat.jhsph.edu/~afrazee/chr22seq.rda'))
  data(gtf_dataframe)
  chr22_processed = seq_gtf(gtf_dataframe, chr22seq)
```

`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,
  num_reps = 10, fraglen = 250, fragsd = 25, readlen = 100,
  error_rate = 0.005, paired = TRUE, reads_per_transcript = 300,
  fold_changes, size = NULL, outdir = ".", write_info = TRUE,
  transcriptid = NULL, seed = NULL, ...)
```

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.
`num_reps`	How many biological replicates should be in each group? If `num_reps` is a single integer, `num_reps` replicates will be simulated in each group. Otherwise, `num_reps` can be a length-2 vector, where `num_reps[1]` and `num_reps[2]` replicates will be simulated in each of the two groups.
`fraglen`	Mean RNA fragment length. Sequences will be read off the end(s) of these fragments.
`fragsd`	Standard deviation of fragment lengths.
`readlen`	Read length.
`error_rate`	Sequencing error rate. Must be between 0 and 1. A uniform error model is assumed.
`paired`	If `TRUE`, paired-end reads are simulated; else single-end reads are simulated.
`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`.
`fold_changes`	Vector of multiplicative fold changes between groups, one entry per transcript in `fasta`. A fold change > 1 means the transcript is overexpressed in the first `num_reps` (or `num_reps[1]`) samples. Fold change < 1 means transcript is overexpressed in the last `num_reps` (or `num_reps[2]`) samples. The change is in the mean number of reads generated from the transcript, between groups.
`size`	the negative binomial `size` parameter (see `NegBinomial`) for the number of reads drawn per transcript. If left blank, defaults to `reads_per_transcript / 3`. Negative binomial variance is mean + mean^2 / size. Can either be left at default, a vector of the same length as number of transcripts in `fasta`, if the two groups should have the same size parameters, or a list with 2 elements, each of which is a vector with length equal to the number of transcripts in `fasta`, which represent the size parameters for each transcript in groups 1 and 2, respectively.
`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.
`write_info`	If `TRUE`, write a file matching transcript IDs to differential expression status into the file `outdir/sim_info.txt`.
`transcriptid`	optional vector of transcript IDs to be written into `sim_info.txt` and used as transcript identifiers in the fasta files. Defaults to `names(readDNAStringSet(fasta))`. This option is useful if default names are very long or contain special characters.
`seed`	Optional seed to set before simulating reads, for reproducibility.
`...`	additional arguments to pass to `seq_gtf` if using `gtf` and `seqpath`

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 and a simulation info file are written to `outdir`.

Examples

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

fastapath = system.file("extdata", "chr22.fa", package="polyester")
numtx = count_transcripts(fastapath)
set.seed(4)
fold_changes = sample(c(0.5, 1, 2), size=numtx,
   prob=c(0.05, 0.9, 0.05), replace=TRUE)
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 = ".", fraglen = 250, fragsd = 25, readlen = 100,
  error_rate = 0.005, paired = TRUE, seed = NULL, ...)
```

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.
`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.
`fraglen`	Mean RNA fragment length. Sequences will be read off the end(s) of these fragments.
`fragsd`	Standard deviation of fragment lengths.
`readlen`	Read length
`error_rate`	Sequencing error rate. Must be between 0 and 1. A uniform error model is assumed.
`paired`	If `TRUE`, paired-end reads are simulated; else single-end reads are simulated.
`seed`	Optional seed to set before simulating reads, for reproducibility.
`...`	Further arguments 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`.

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)
```

`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)
```

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.

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)
```

Index