sagenhaft

October 5, 2010

`error.correction` *Estimate sequencing errors and compute corrected counts*

Description

These functions are used to compute sequencing error correction in a library. They are automatically called when extracting tags from sequences and therefore usually do not have to be called directly.

Usage

```
estimate.errors.mean(lib)
compute.sequence.neighbors(tags, taglength=10, quality.scores=NULL,
                           output="character")
em.estimate.error.given(lib, maxstep=50, ...)
```

Arguments

`lib`	A sage library object
`tags`	A character vector or numeric vector containing tags
`taglength`	length of tag
`quality.scores`	A matrix containing base quality scores as -10 log10 Pe
`maxstep`	iterations of EM algorithm
`output`	Output type for `compute.sequence.neighbors`, either character or numeric
`...`	Other arguments ignored.

Author(s)

Tim Beissbarth

References

http://tagcalling.mbgproject.org

See Also

`extract.lib`, `sage.library`

Examples

```
library(sagenhaft)
B6Hypo <-read.sage.library(system.file("data/B6HypothalHFI.sage",
                           package="sagenhaft"))
E15post <- read.sage.library(system.file("data/E15postHFI.sage",
                             package="sagenhaft"))
testlib <- combine.libs(B6Hypo, E15post)
testlib <- estimate.errors.mean(testlib)
testlib <- em.estimate.error.given(testlib)
tagneighbors <- compute.sequence.neighbors(testlib$seqs[,"seq"], 10,
                          testlib$seqs[,paste("q", 1:10, sep="")])
```

`extract.lib` *Functions for SAGE library extraction*

Description

Functions to extract the tags in a library from sequences or base-caller output.

Usage

```
extract.lib.from.zip(zipfile, libname=sub(".zip","",basename(zipfile)),
                     ...)
extract.lib.from.directory(dirname, libname=basename(dirname),
                           pattern, ...)
extract.library.tags(filelist, base.caller.format="phd",
                     remove.duplicate.ditags=TRUE,
                     remove.N=FALSE, remove.low.quality=10,
                     taglength=10, min.ditag.length=(2*taglength-2),
                     max.ditag.length=(2*taglength+4),
                     cut.site="catg", default.quality=NA, verbose=TRUE,
                     ...)
reestimate.lib.from.tagcounts(tagcounts, libname, default.quality=20, ...)
compute.unique.tags(lib)
combine.libs(..., artifacts=c("Linker", "Ribosomal", "Mitochondrial"))
remove.sage.artifacts(lib,
                      artifacts=c("Linker","Ribosomal","Mitochondrial"),
                      ...)
read.phd.file(file)
read.seq.qual.filepair(file, default.quality=NA)
extract.ditags(sequence, taglength=10, filename=NA,
               min.ditag.length=(2*taglength-2),
               max.ditag.length=(2*taglength+4), cut.site="catg")
```

Arguments

`zipfile,dirname`	Name of a ZIP file or a directory that contains base-caller output files
`libname`	`libname` a character string to be assigned as library name
`pattern`	Regular expression to specify pattern for the files that will be read

filelist	List of files to be read
base.caller.format	base.caller.format can be "phd" or "seq" or a character vector of the length of the filelist
remove.duplicate.ditags	Remove duplicate ditags. TRUE or FALSE
remove.N	Remove all tags that contain N. TRUE or FALSE
remove.low.quality	Remove all tags with an average quality score of less than remove.low.quality. Skipped if < 0
taglength	Length of tags. Usually 10 or 17
min.ditag.length,max.ditag.length	Minimum and maximum length for ditags
cut.site	Restriction enzyme cut site. Usually CATG
verbose	Display information during process
lib	Library object
file,filename	Character string indicating file name
default.quality	Quality value to use on sequences, if quality files are missing
sequence	Construct containing sequence and quality values returned by read.phd.file or read.seq.qual.filepair
artifacts	Types of artificially generated tags to remove.
...	Arguments passed on to extraction functions.
tagcounts	Tagcounts from library. Integer Vecotor with Tag sequences as names.

Details

The functions extract.lib.from.zip or extract.lib.from.directory should be used to extract the SAGE TAGS from the sequences of a library, the sequences need to be provided by the output files from the base caller software either in a ZIP archive or in a directory. These are usually the only functions that should directly be called by the user. The other functions are called by these and should only be used directly by experienced users to get more direct control over the process. Most arguments are passed on and can be specified in the high level functions. Zipfilenames must be specified using relative pathnames!

Value

lib returns an SAGE library object.

Author(s)

Tim Beissbarth

References

http://tagcalling.mbgproject.org

See Also

sage.library, error.correction

Examples

```
#library(sagenhaft)
#file.copy(system.file("data/E15postHFI.zip",package="sagenhaft"),
#          "E15postHFI.zip")
#E15post<-extract.lib.from.zip("E15postHFI.zip", taglength=10,
#                              min.ditag.length=20, max.ditag.length=24)
#E15post
```

`sage.library` *Class sage.library*

Description

The SAGE library class contains all the data and annotation for a SAGE library. It can contain two data.frames.

Usage

```
read.sage.library(file)
write.sage.library(x, file=paste(x$libname, "sage", sep="."),
                   what="complete")
```

Arguments

`x`	A sage library object
`file`	File name to read or write to
`what`	"complete", read complete librarary tags and sequences; "tags", read only tags and counts

Details

SAGE library objects consists of one or two data.frames. The data.frame "tags" contains all the unique tags in the library and its counts. The data.frame "seqs" contains all the individual tag sequences and associated quality values. `read.sage.library` and `write.sage.library` are utility functions to read and write SAGE libraries.

Author(s)

Tim Beissbarth

References

http://tagcalling.mbgproject.org

See Also

`extract.lib`

Examples

```
library(sagenhaft)
E15postHFI <- read.sage.library(system.file("data/E15postHFI.sage",
                                package="sagenhaft"))
E15postHFI
```

`sage.library.comparison`
Class sage.library.comparison

Description

Class for storing the data of a pairwise comparison between two SAGE libraries.

Usage

```
read.sage.library.comparison(file)
write.sage.library.comparison(x, file=paste(x$name, "sagecomp", sep="."))
compare.lib.pair(lib1, lib2)
```

Arguments

`x,lib1,lib2`	A sage library object
`file`	File name to read or write to

Details

SAGE library comparison objects consists of one data.frames. It stores a A and an M value which are the log2 average expression and log2 ratio, respectively. It also has a column for the resulting p.values from `sage.test`. `read.sage.library.comparison` and `write.sage.library.comparison` are utility functions to read and write SAGE library comparisons. `compare.lib.pair` can be used to generate SAGE library comparisons.

Author(s)

Tim Beissbarth

References

http://tagcalling.mbgproject.org

See Also

`sage.test`

Examples

```
library(sagenhaft)
B6Hypo <- read.sage.library(system.file("data/B6HypothalHFI.sage",
                            package="sagenhaft"))
E15post <- read.sage.library(system.file("data/E15postHFI.sage",
                             package="sagenhaft"))
libcomp <- compare.lib.pair(B6Hypo, E15post)
plot(libcomp)
libcomp
```

`sage.test` *Compare Two SAGE Libraries*

Description

Compute p-values for differential expression for each tag between two SAGE libraries.

Usage

```
sage.test(x, y, n1=sum(x), n2=sum(y))
```

Arguments

`x`	integer vector giving counts in first library. Non-integer values are rounded to the nearest integer.
`y`	integer vector giving counts in second library. Non-integer values are rounded to the nearest integer.
`n1`	total number of tags in first library. Non-integer values are rounded to the nearest integer.
`n2`	total number of tags in second library. Non-integer values are rounded to the nearest integer.

Details

This function uses a binomial approximation to the Fisher Exact test for each tag. The approximation is accurate when `n1` and `n2` are large and `x` and `y` are small in comparison.

Value

Numeric vector of p-values.

Author(s)

Gordon Smyth

See Also

`fisher.test`

Examples

```
library(sagenhaft)
sage.test(c(0,5,10),c(0,30,50),n1=10000,n2=15000)
#  Exact equivalents
fisher.test(matrix(c(0,0,10000-0,15000-0),2,2))$p.value
fisher.test(matrix(c(5,30,10000-5,15000-30),2,2))$p.value
fisher.test(matrix(c(10,50,10000-10,15000-50),2,2))$p.value
```

---

sage.utilities *Utilities*

Description

Different utilities to use with SAGE data.

Usage

```
tagnum2tagmatrix(tags, length)
tagmatrix2tagnum(tags, length=ncol(tags))
tagnum2tagsequence(tags, length)
tagsequence2tagnum(tags, length)
revcomp(seq)
```

Arguments

tags	integer or character vector giving SAGE tags.
length	Length of SAGE tags.
seq	Character vector or list of sequences.
...	SAGE library objects.

Details

These functions are utility functions used in SAGE tag extraction, e.g. to convert SAGE tag sequences to numeric values, i.e. base 4 for efficient storage and handling, and to reverse complement sequences.

Author(s)

Tim Beissbarth

Examples

```
library(sagenhaft)
tags <- c("aaa", "ttt", "ccc")
tagsnumeric <- tagsequence2tagnum(tags, 3)
tagsmatrix <- tagnum2tagmatrix(tagsnumeric, 3)
tags <- tagnum2tagsequence(tagmatrix2tagnum(tagsmatrix, 3), 3)
revcomp(tags)
```

`sagelibrary.simulate` *Simulate SAGE libraries*

Description

Function to simulate SAGE libraries with sequencing errors.

Usage

```
sagelibrary.simulate(taglength = 4, lambda = 1000, mean.error = 0.01,
                 error.sd = 1, withintagerror.sd = 0.2,
                 ngenes = min(4^taglength, 1e+05), base.lib = NULL,
                 libseed = -1, ...)
```

Arguments

`taglength`	Tag length for library.
`lambda`	Aproximate size of library.
`mean.error`	Mean amount of sequencing errors.
`error.sd`	Standard deviation for sequencing errors.
`withintagerror.sd`	Standard deviation for sequencing errors within tags.
`ngenes`	Number of genes to generate tags from.
`base.lib`	Simulate library based on tags in other lib and create variations.
`libseed`	Seed for random number generator.
`...`	Arguments passed to em.estimate.

Details

We set the number of possible transcripts and assign a random SAGE tag to each of them out of all 4\^taglength possible SAGE tags. For each SAGE tag a random proportion p within the library is generated from a log-normal distribution, and the proportions are then adjusted to have a sum of 1. The true counts of a tag are simulated by sampling from Poisson distributions with parameters p lambda, where p is the proportion of the tag in the library and lambda is a parameter for setting the size of the library. The simulation of the sequencing errors is done on each individual occurrence of a tag sequence. For each tag sequence a mean sequencing quality value is generated from a log-normal distribution. The individual quality values for each base are then generated from log-normal distributions with means equal to the simulated sequencing quality values for the tag sequences. We have noticed that with experimentally generated data the within tag sequence variation of sequencing quality values is usually about 1/5 of the between tag sequence variation. From each true tag sequence one observed tag sequence is generated using the simulated quality values of the true sequence as the multinomial probabilities, i.e. replacing each base with either one of the 3 other bases with the probability specified by the sequencing quality value of that base. The counts of these generated tags are then summed to represent the observed tags. When generating several simulated libraries for comparisons, we use the same proportions of the genes for all libraries, replacing up to 1/3 of the proportions by proportions with a known differential factor.

Author(s)

Tim Beissbarth

References

http://tagcalling.mbgproject.org

See Also

sage.library, error.correction

Examples

```
library(sagenhaft)
testlib1 <- sagelibrary.simulate(taglength=10, lambda=10000,
                                 mean.error=0.01)
testlib2 <- sagelibrary.simulate(taglength=10, lambda=20000,
                                 mean.error=0.02, base.lib=testlib1)
testlib3 <- sagelibrary.simulate(taglength=10, lambda=10000,
                                 mean.error=0.01, libseed=testlib1$seed)
```

Index