Package 'ShortRead'

December 27, 2024

Type Package

Title FASTQ input and manipulation

Version 1.65.0

Description This package implements sampling, iteration, and input of FASTQ files. The package includes functions for filtering and trimming reads, and for generating a quality assessment report. Data are represented as DNAStringSet-derived objects, and easily manipulated for a diversity of purposes. The package also contains legacy support for early single-end, ungapped alignment formats.

License Artistic-2.0

LazyLoad yes

Depends BiocGenerics (>= 0.23.3), BiocParallel, Biostrings (>= 2.47.6), Rsamtools (>= 1.31.2), GenomicAlignments (>= 1.15.6)

Imports Biobase, S4Vectors (>= 0.17.25), IRanges (>= 2.13.12), GenomeInfoDb (>= 1.15.2), GenomicRanges (>= 1.31.8), pwalign, hwriter, methods, lattice, latticeExtra,

Suggests BiocStyle, RUnit, biomaRt, GenomicFeatures, yeastNagalakshmi, knitr

LinkingTo S4Vectors, IRanges, XVector, Biostrings, Rhtslib

biocViews DataImport, Sequencing, QualityControl

URL https://bioconductor.org/packages/ShortRead, https://github.com/Bioconductor/ShortRead, https://support.bioconductor.org/tag/ShortRead

BugReports https://github.com/Bioconductor/ShortRead/issues

VignetteBuilder knitr

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

git_branch devel

git_last_commit ad4e1df

git_last_commit_date 2024-10-29

Repository Bioconductor 3.21

Date/Publication 2024-12-26

Author Bioconductor Package Maintainer [cre],
Martin Morgan [aut],
Michael Lawrence [ctb],
Simon Anders [ctb],
Rohit Satyam [ctb] (Converted Overview.Rnw vignette from Sweave to RMarkdown / HTML.),
J Wokaty [ctb]

Maintainer Bioconductor Package Maintainer <maintainer@bioconductor.org>

Contents

`ShortReadBase-package` *FASTQ input and manipulation.*

Description

This package implements sampling, iteration, and input of FASTQ files. The package includes functions for filtering and trimming reads, and for generating a quality assessment report. Data are represented as DNAStringSet-derived objects, and easily manipulated for a diversity of purposes. The package also contains legacy support for early single-end, ungapped alignment formats.

Details

See `packageDescription('ShortRead')`

Author(s)

Maintainer: Martin Morgan <mtmorgan@fhcrc.org>

.QA-class *Virtual class for representing quality assessment results*

Description

Classes derived from .QA-class represent results of quality assurance analyses. Details of derived class structure are found on the help pages of the derived classes.

Objects from the Class

Objects from the class are created by ShortRead functions, in particular qa.

Extends

Class ".ShortReadBase", directly.

Methods

Methods defined on this class include:

rbind signature(...="list"): rbind data frame objects in .... All objects of ... must be of the same class; the return value is an instance of that class.

show signature(object = "SolexaExportQA"): Display an overview of the object contents.

Author(s)

Martin Morgan <mtmmorgan@fhcrc.org>

See Also

Specific classes derived from .QA

Examples

```
getClass(".QA", where=getNamespace("ShortRead"))
```

accessors *(Legacy) Accessors for ShortRead classes*

Description

These functions and generics define 'accessors' (to get and set values) for objects in the **ShortRead** package; methods defined in other packages may have additional meaning.

Usage

```
## SRVector
vclass(object, ...)
## AlignedRead
chromosome(object, ...)
position(object, ...)
alignQuality(object, ...)
alignData(object, ...)
## Solexa
experimentPath(object, ...)
dataPath(object, ...)
scanPath(object, ...)
imageAnalysisPath(object, ...)
baseCallPath(object, ...)
analysisPath(object, ...)
## SolexaSet
solexaPath(object, ...)
laneDescription(object, ...)
laneNames(object, ...)
```

Arguments

`object`	An object derived from class `ShortRead`. See help pages for individual objects, e.g., `ShortReadQ`. The default is to extract the contents of a slot of the corresponding name (e.g., slot `sread`) from `object`.
`...`	Additional arguments passed to the accessor. The default definitions do not make use of additional arguments.

Value

Usually, the value of the corresponding slot, or other simple content described on the help page of `object`.

Author(s)

Martin Morgan

Examples

```
sp <- SolexaPath(system.file('extdata', package='ShortRead'))
experimentPath(sp)
basename(analysisPath(sp))
```

`AlignedDataFrame` *(Legacy) AlignedDataFrame constructor*

Description

Construct an `AlignedDataFrame` from a data frame and its metadata

Usage

```
AlignedDataFrame(data, metadata, nrow = nrow(data))
```

Arguments

`data`	A data frame containing alignment information.
`metadata`	A data frame describing the columns of `data`, and with number of rows of `metadata` corresponding to number of columns of `data`. . The data frame must contain a column `labelDescription` providing a verbose description of each column of `data`.
`nrow`	An optional argument, to be used when `data` is not provided, to construct an AlignedDataFrame with the specified number of rows.

Value

An object of `AlignedDataFrame`.

Author(s)

Martin Morgan <mtmorgan@fhcrc.org>

`AlignedDataFrame-class` *(Legacy) "AlignedDataFrame" representing alignment annotations as a data frame*

Description

This class extends `AnnotatedDataFrame`. It is a data frame and associated metadata (describing the columns of the data frame). The main purpose of this class is to contain alignment data in addition to the central information of `AlignedRead`.

Objects from the Class

Objects from the class are created by calls to the AlignedDataFrame function.

Slots

data: Object of class "data.frame" containing the data. See AnnotatedDataFrame for details.

varMetadata: Object of class "data.frame" describing columns of data. See AnnotatedDataFrame for details.

dimLabels: Object of class character describing the dimensions of the AnnotatedDataFrame. Used internally; see AnnotatedDataFrame for details.

.__classVersion__: Object of class "Versions" describing the version of this object. Used internally; see AnnotatedDataFrame for details.

Extends

Class "AnnotatedDataFrame", directly. Class "Versioned", by class "AnnotatedDataFrame", distance 2.

Methods

This class inherits methods pData (to retrieve the underlying data frame) and varMetadata (to retrieve the metadata) from AnnotatedDataFrame.

Additional methods include:

append signature(x = "AlignedDataFrame", values = "AlignedDataFrame"): append values after x. varMetadata of x and y must be identical; pData and varMetadata are appended using rbind.

Author(s)

Martin Morgan <mtmorgan@fhcrc.org>

See Also

AnnotatedDataFrame

AlignedRead *(Legacy) Construct objects of class "AlignedRead"*

Description

This function constructs objects of AlignedRead. It will often be more convenient to create AlignedRead objects using parsers such as readAligned.

Usage

```
AlignedRead(sread, id, quality, chromosome, position, strand,
            alignQuality,
            alignData = AlignedDataFrame(nrow = length(sread)))
```

Arguments

sread	An object of class DNAStringSet, containing the DNA sequences of the short reads.
id	An object of class BStringSet, containing the identifiers of the short reads. This object is the same length as sread.
quality	An object of class BStringSet, containing the ASCII-encoded quality scores of the short reads. This object is the same length as sread.
chromosome	A factor describing the particular sequence within a set of target sequences (e.g. chromosomes in a genome assembly) to which each short read aligns.
position	A integer vector describing the (base pair) position at which each short read begins its alignment.
strand	A factor describing the strand to which the short read aligns.
alignQuality	A numeric vector describing the alignment quality.
alignData	An AlignedDataFrame with number of rows equal to the length of sread, containing additional information about alignments.

Value

An object of class AlignedRead.

Author(s)

Martin Morgan <mtmorgan@fhcrc.org>

See Also

AlignedRead.

AlignedRead-class *(Legacy) "AlignedRead" class for aligned short reads*

Description

This class represents and manipulates reads and their genomic alignments. Alignment information includes genomic position, strand, quality, and other data.

Objects from the Class

Objects of this class can be created from a call to the AlignedRead constructor, or more typically by parsing appropriate files (e.g., readAligned).

Slots

`chromosome` Object of class `"factor"` the particular sequence within a set of target sequences (e.g. chromosomes in a genome assembly) to which each short read aligns.

`position` Object of class `"integer"` the (base-pair) position in the genome to which the read is aligned. AlignedRead objects created by readAligned use 1-based indexing, with alignemnts reported in 'left-most' coordinates, as described in the vignette.

`strand` Object of class `"factor"` the strand of the alignment.

`alignQuality` Object of class `"numeric"` representing an alignment quality score.

`alignData` Object of class `"AlignedDataFrame"` additional alignment information.

`quality` Object of class `"BStringSet"` representing base-call read quality scores.

`sread` Object of class `"DNAStringSet"` DNA sequence of the read.

`id` Object of class `"BStringSet"` read identifier.

Extends

Class `"ShortReadQ"`, directly. Class `"ShortRead"`, by class "ShortReadQ", distance 2. Class `".ShortReadBase"`, by class "ShortReadQ", distance 3.

Methods

See `accessors` for additional functions to access slot content, and `ShortReadQ`, `ShortRead` for inherited methods. Additional methods include:

[`signature(x = "AlignedRead", i = "ANY", j = "missing")`: This method creates a new `AlignedRead` object containing only those reads indexed by `i`. `chromosome` is recoded to contain only those levels in the new subset.

append `signature(x = "AlignedRead", values = "AlignedRead")`: append `values` after `x`. `chromosome` and `strand` must be factors with the same levels. See methods for `ShortReadQ`, `AlignedDataFrame` for details of how these components of `x` and `y` are appended.

coerce `signature(from = "PairwiseAlignments", to = "AlignedRead")`:

`signature(from = "AlignedRead", to = "IntegerRangesList")`: `signature(from = "AlignedRead", to = "GRanges")`: `signature(from = "AlignedRead", to = "GAlignments")`: `signature(from = "AlignedRead", to = "GappedReads")`:

Invoke these methods with, e.g., `as(from, "AlignedRead")` to coerce objects of class `from` to class `"AlignedRead"`.

Coercion from AlignedRead to IntegerRangesList or GRanges assumes that `position(from)` uses a 'leftmost' (see `coverage` on this page) coordinate system. Since IntegerRangesList objects cannot store `NA` values, reads with `NA` in the `position`, `width`, `chromosome` or (in the case of GRanges) `strand` vectors are dropped.

chromosome `signature(object = "AlignedRead")`: access the chromosome slot of `object`.

position `signature(object = "AlignedRead")`: access the position slot of `object`.

strand `signature(object = "AlignedRead")`: access the strand slot of `object`.

coverage `signature(x = "AlignedRead", shift = 0L, width = NULL, weight = 1L, ..., coords = c("leftmost", "fiveprime"), extend=0L)`:

Calculate coverage across reads present in `x`.

`shift` must be either `0L` or a named integer vector with names including all `levels(chromosome(x))`. It specifies how the reads in `x` should be (horizontally) shifted *before* the coverage is computed.

`width` must be either `NULL` or a named vector of non-negative integers with names including all `levels(chromosome(x))`. In the latter case, it specifies for each chromosome the end of the chromosome region over which coverage is to be calculated *after* the reads have been shifted. Note that this region always starts at chromosome position 1. If `width` is `NULL`, it ends at the rightmost chromosome position covered by at least one read.

`weight` must be `1L` for now (weighting the reads is not supported yet, sorry).

`coords` specifies the coordinate system used to record position. Both systems number base pairs from left to right on the 5' strand. `leftmost` indicates the eland convention, where `position(x)` is the left-most (minimum) base pair, regardless of strand. `fiveprime` is the MAQ convention, where `position(x)` is the coordinate of the 5' end of the aligned read.

`extend` indicates the number of base pairs to extend the read. Extension is in the 3' direction, measured from the 3' end of the aligned read.

The return value of `coverage` is a `SimpleRleList` object.

%in% `signature(x = "AlignedRead", table = "IntegerRangesList")`:

Return a length(x) logical vector indicating whether the chromosome, position, and width of `x` overlap with ranges in `table`. Reads for which `chromosome()`, `position()`, or `width()` return `NA` *never* overlap with `table`. This function assumes that positions are in 'leftmost' coordinates, as defined in `coverage`.

srorder `signature(x = "AlignedRead", ..., withSread=TRUE)`:

srrank `signature(x = "AlignedRead", ..., withSread=TRUE)`:

srsort `signature(x = "AlignedRead", ..., withSread=TRUE)`:

srduplicated `signature(x = "AlignedRead", ..., withSread=TRUE)`:

Order, rank, sort, and find duplicates in `AlignedRead` objects. Reads are sorted by `chromosome`, `strand`, `position`, and then (if `withSread=TRUE`) `sread`; less fine-grained sorting can be accomplished with, e.g., `x[srorder(sread(x))]`. `srduplicated` behaves like `duplicated`, i.e., the first copy of a duplicate is `FALSE` while the remaining copies are `TRUE`.

show `signature(object = "AlignedRead")`: provide a compact display of the `AlignedRead` content.

detail `signature(x = "AlignedRead")`: display `alignData` in more detail.

Author(s)

Martin Morgan <mtmorgan@fhcrc.org>

See Also

`readAligned`

Examples

```
showMethods(class="AlignedRead", where=getNamespace("ShortRead"))
dirPath <- system.file('extdata', 'maq', package='ShortRead')
(aln <- readAligned(dirPath, 'out.aln.1.txt', type="MAQMapview"))
coverage(aln)[[1]]
cvg <- coverage(aln, shift=c(ChrA=10L))
## remove 0 coverage on left ends
ltrim0 <- function(x) {
    i <- !cumprod(runValue(x) == 0)
    Rle(runValue(x)[i], runLength(x)[i])
}
endoapply(cvg, ltrim0)
## demonstration of show() and detail() methods
show(aln)
detail(aln)
```

`alphabetByCycle` *Summarize nucleotide, amino acid, or quality scores by cycle*

Description

`alphabetByCycle` summarizes nucleotides, amino acid, or qualities by cycle, e.g., returning the number of occurrences of each nucleotide `A`, `T`, `G`, `C` across all reads from 36 cycles of a Solexa lane.

Usage

```
alphabetByCycle(stringSet, alphabet, ...)
```

Arguments

`stringSet`	A R object representing the collection of reads, amino acid sequences, or quality scores, to be summarized.
`alphabet`	The alphabet (character vector of length 1 strings) from which the sequences in `stringSet` are composed. Methods often define an appropriate alphabet, so that the user does not have to provide one.
`...`	Additional arguments, perhaps used by methods defined on this generic.

Details

The default method requires that `stringSet` extends the `XStringSet` class of **Biostrings**.

The following method is defined, in addition to methods described in class-specific documentation:

alphabetByCycle `signature(stringSet = "BStringSet")`: this method uses an alphabet spanning all ASCII characters, codes `1:255`.

Value

A matrix with number of rows equal to the length of `alphabet` and columns equal to the maximum width of reads or quality scores in the string set. Entries in the matrix are the number of times, over all reads of the set, that the corresponding letter of the alphabet (row) appeared at the specified cycle (column).

Author(s)

Martin Morgan

See Also

The IUPAC alphabet in Biostrings.

http://www.bioperl.org/wiki/FASTQ_sequence_format for the BioPerl definition of fastq.

Solexa documentation 'Data analysis - documentation : Pipeline output and visualisation'.

Examples

```
showMethods("alphabetByCycle")

sp <- SolexaPath(system.file('extdata', package='ShortRead'))
rfq <- readFastq(analysisPath(sp), pattern="s_1_sequence.txt")
alphabetByCycle(sread(rfq))

abcq <- alphabetByCycle(quality(rfq))
dim(abcq)
## 'high' scores, first and last cycles
abcq[64:94,c(1:5, 32:36)]
```

`alphabetScore` *Efficiently calculate the sum of quality scores across bases*

Description

This generic takes a `QualityScore` or `PhredQuality` object and calculates, for each read, the sum of the encoded nucleotide probabilities.

Usage

```
alphabetScore(object, ...)
```

Arguments

`object`	An object of class `QualityScore`.
`...`	Additional arguments, currently unused.

Value

A vector of numeric values of length equal to the length of `object`.

Author(s)

Martin Morgan <mtmorgan@fhcrc.org>

`BowtieQA-class` *(Legacy) Quality assessment summaries from Bowtie files*

Description

This class contains a list-like structure with summary descriptions derived from visiting one or more Bowtie files.

Objects from the Class

Objects of the class are usually produced by a `qa` method, with the argument `type="Bowtie"`.

Slots

`.srlist`: Object of class `"list"`, containing data frames or lists of data frames summarizing the results of qa.

Extends

Class `"SRList"`, directly. Class `".QA"`, directly. Class `".SRUtil"`, by class "SRList", distance 2. Class `".ShortReadBase"`, by class ".QA", distance 2.

Methods

Accessor methods are inherited from the `SRList` class.

report `signature(x="BowtieQA", ..., dest=tempfile(), type="html")`: produces an html file summarizing the QA results.

Author(s)

Martin Morgan <mtmorgan@fhcrc.org>

See Also

`qa`.

Examples

```
showClass("BowtieQA")
```

clean *Remove sequences with ambiguous nucleotides from short read classes*

Description

Short reads may contain ambiguous base calls (i.e., IUPAC symbols different from A, T, G, C). This generic removes all sequences containing 1 or more ambiguous bases.

Usage

```
clean(object, ...)
```

Arguments

object	An object for which clean methods exist; see below to discover these methods.
...	Additional arguments, perhaps used by methods.

Details

The following method is defined, in addition to methods described in class-specific documentation:

clean `signature(x = "DNAStringSet")`: Remove all sequences containing non-base (A, C, G, T) IUPAC symbols.

Value

An instance of `class(object)`, containing only sequences with non-redundant nucleotides.

Author(s)

Martin Morgan <mtmorgan@fhcrc.org>

Examples

```
showMethods('clean')
```

countLines — *Count lines in all (text) files in a directory whose file name matches a pattern*

Description

countLines visits all files in a directory path dirPath whose base (i.e., file) name matches pattern. Lines in the file are counted as the number of new line characters.

Usage

```
countLines(dirPath, pattern=character(0), ..., useFullName=FALSE)
```

Arguments

dirPath	A character vector (or other object; see methods defined on this generic) giving the directory path (relative or absolute) of files whose lines are to be counted.
pattern	The (grep-style) pattern describing files whose lines are to be counted. The default (character(0)) results in line counts for all files in the directory.
...	Additional arguments, passed internally to list.files. See list.files.
useFullName	A logical(1) indicating whether elements of the returned vector should be named with the base (file) name (default; useFullName=FALSE) or the full path name (useFullName=TRUE).

Value

A named integer vector of line counts. Names are paths to the files whose lines have been counted, excluding dirPath.

Author(s)

Martin Morgan

Examples

```
sp <- SolexaPath(system.file('extdata', package='ShortRead'))
countLines(analysisPath(sp))
countLines(experimentPath(sp), recursive=TRUE)
countLines(experimentPath(sp), recursive=TRUE, useFullName=TRUE)
```

deprecated *Deprecated and defunct functions*

Description

These functions were introduced but are now deprecated or defunct.

Details

Defunct functions:

- `srapply`. Use the BiocParallel package instead.
- `readAligned,BamFile-method`. Use the GenomicAlignments package instead.
- `basePath()`

dustyScore *Summarize low-complexity sequences*

Description

`dustyScore` identifies low-complexity sequences, in a manner inspired by the `dust` implementation in `BLAST`.

Usage

```
dustyScore(x, batchSize=NA, ...)
```

Arguments

`x`	A `DNAStringSet` object, or object derived from `ShortRead`, containing a collection of reads to be summarized.
`batchSize`	`NA` or an `integer(1)` vector indicating the maximum number of reads to be processed at any one time.
`...`	Additional arguments, not currently used.

Details

The following methods are defined:

dustyScore `signature(x = "DNAStringSet")`: operating on an object derived from class `DNAStringSet`.

dustyScore `signature(x = "ShortRead")`: operating on the `sread` of an object derived from class `ShortRead`.

The dust-like calculations used here are as implemented at https://stat.ethz.ch/pipermail/bioc-sig-sequencing/2009-February/000170.html. Scores range from 0 (all triplets unique) to the square of the width of the longest sequence (poly-A, -C, -G, or -T).

The `batchSize` argument can be used to reduce the memory requirements of the algorithm by processing the `x` argument in batches of the specified size. Smaller batch sizes use less memory, but are computationally less efficient.

Value

A vector of numeric scores, with length equal to the length of `x`.

Author(s)

Herve Pages (code); Martin Morgan

References

Morgulis, Getz, Schaffer and Agarwala, 2006. WindowMasker: window-based masker for sequenced genomes, Bioinformatics 22: 134-141.

See Also

The WindowMasker supplement defining `dust` ftp://ftp.ncbi.nlm.nih.gov/pub/agarwala/windowmasker/windowmasker_suppl.pdf

Examples

```
sp <- SolexaPath(system.file('extdata', package='ShortRead'))
rfq <- readFastq(analysisPath(sp), pattern="s_1_sequence.txt")
range(dustyScore(rfq))
```

ExperimentPath-class *(Legacy) "ExperimentPath" class representing a file hierarchy of data files*

Description

Short read technologies often produce a hierarchy of output files. The content of the hierarchy varies. This class represents the root of the file hierarchy. Specific classes (e.g., `SolexaPath`) represent different technologies.

Objects from the Class

Objects from the class are created by calls to the constructor:

`ExperimentPath(experimentPath)`

experimentPath `character(1)` object pointing to the top-level directory of the experiment; see specific technology classes for additional detail.

verbose=FALSE (optional) logical vector which, when TRUE results in warnings if paths do not exist.

All paths must be fully-specified.

Slots

ExperimentPath has one slot, containing a fully specified path to the corresponding directory (described above).

basePath See above.

The slot is accessed with experimentPath.

Extends

Class ".ShortReadBase", directly.

Methods

Methods include:

show signature(object = "ExperimentPath"): briefly summarize the file paths of object.

detail signature(x = "ExperimentPath"): summarize file paths of x.

Author(s)

Michael Lawrence

Examples

```
showClass("ExperimentPath")
```

FastqFile-class *Sampling and streaming records from fastq files*

Description

FastqFile represents a path and connection to a fastq file. FastqFileList is a list of such connections.

FastqSampler draws a subsample from a fastq file. yield is the method used to extract the sample from the FastqSampler instance; a short illustration is in the example below. FastqSamplerList is a list of FastqSampler elements.

FastqStreamer draws successive subsets from a fastq file, a short illustration is in the example below. FastqStreamerList is a list of FastqStreamer elements.

Usage

```
## FastqFile and FastqFileList
FastqFile(con, ...)
FastqFileList(..., class="FastqFile")
## S3 method for class 'ShortReadFile'
open(con, ...)
## S3 method for class 'ShortReadFile'
close(con, ...)
## S4 method for signature 'FastqFile'
readFastq(dirPath, pattern=character(), ...)
## S4 method for signature 'FastqFile'
countFastq(dirPath, pattern=character(), ...)

## FastqSampler and FastqStreamer
FastqSampler(con, n=1e6, readerBlockSize=1e8, verbose=FALSE,
    ordered = FALSE)
FastqSamplerList(..., n=1e6, readerBlockSize=1e8, verbose=FALSE,
    ordered = FALSE)
FastqStreamer(con, n, readerBlockSize=1e8, verbose=FALSE)
FastqStreamerList(..., n, readerBlockSize=1e8, verbose=FALSE)
yield(x, ...)
```

Arguments

con, dirPath	A character string naming a connection, or (for con) an R connection (e.g., file, gzfile).
n	For FastqSampler, the size of the sample (number of records) to be drawn. For FastqStreamer a numeric(1) (set to 1e6 when n is missing) providing the number of successive records to be returned on each yield, or an IRanges-class delimiting the (1-based) indicies of records returned by each yield; entries in n must have non-zero width and must not overlap.
readerBlockSize	The number of bytes or characters to be read at one time; smaller readerBlockSize reduces memory requirements but is less efficient.
verbose	Display progress.
ordered	logical(1) indicating whether sampled reads should be returned in the same order as they were encountered in the file.
x	An instance from the FastqSampler or FastqStreamer class.
...	Additional arguments. For FastqFileList, FastqSamplerList, or FastqStreamerList, this can either be a single character vector of paths to fastq files, or several instances of the corresponding FastqFile, FastqSampler, or FastqStreamer objects.
pattern	Ignored.
class	For developer use, to specify the underlying class contained in the FastqFileList.

Objects from the class

Available classes include:

`FastqFile` A file path and connection to a fastq file.

`FastqFileList` A list of `FastqFile` instances.

`FastqSampler` Uniformly sample records from a fastq file.

`FastqStreamer` Iterate over a fastq file, returning successive parts of the file.

Methods

The following methods are available to users:

`readFastq,FastqFile-method`**:** see also ?`readFastq`.

`writeFastq,ShortReadQ,FastqFile-method`**:** see also ?`writeFastq`, ?"`writeFastq,ShortReadQ,FastqFile-method`"

`countFastq,FastqFile-method`**:** see also ?`countFastq`.

`yield`**:** Draw a single sample from the instance. Operationally this requires that the underlying data (e.g., file) represented by the `Sampler` instance be visited; this may be time consuming.

Note

`FastqSampler` and `FastqStreamer` use OpenMP threads, when available, during creation of the return value. This may cause the OpenMP implementation 'libgomp' to produce an error, if these functions are called in a parallel R process, e.g.:

```
libgomp: Thread creation failed: Resource temporarily unavailable
```

A solution is to precede problematic code with the following code snippet, to disable OpenMP multi-threading:

```
nthreads <- .Call(ShortRead:::.set_omp_threads, 1L)
on.exit(.Call(ShortRead:::.set_omp_threads, nthreads))
```

See Also

`readFastq`, `writeFastq`, `countFastq`, `yield`.

Examples

```
sp <- SolexaPath(system.file('extdata', package='ShortRead'))
fl <- file.path(analysisPath(sp), "s_1_sequence.txt")

f <- FastqFile(fl)
rfq <- readFastq(f)
close(f)

f <- FastqSampler(fl, 50)
```

```
yield(f)    # sample of size n=50
yield(f)    # independent sample of size 50
close(f)

## Return sample as ordered in original file
f <- FastqSampler(fl, 50, ordered=TRUE)
yield(f)
close(f)

f <- FastqStreamer(fl, 50)
yield(f)    # records 1 to 50
yield(f)    # records 51 to 100
close(f)

## iterating over an entire file
f <- FastqStreamer(fl, 50)
while (length(fq <- yield(f))) {
    ## do work here
    print(length(fq))
}
close(f)

## iterating over IRanges
rng <- IRanges(c(50, 100, 200), width=10:8)
f <- FastqStreamer(fl, rng)
while (length(fq <- yield(f))) {
    print(length(fq))
}
close(f)

## Internal fields, methods, and help; for developers
ShortRead:::.FastqSampler_g$methods()
ShortRead:::.FastqSampler_g$fields()
ShortRead:::.FastqSampler_g$help("yield")
```

`filterFastq` *Filter fastq from one file to another*

Description

`filterFastq` filters reads from source to destination file(s) applying a filter to reads in each file. The filter can be a function or FilterRules instance; operations are done in a memory-efficient manner.

Usage

```
filterFastq(files, destinations, ..., filter = FilterRules(),
    compress=TRUE, yieldSize = 1000000L)
```

Arguments

files	a character vector of valid file paths.
destinations	a character vector of destinations, recycled to be the same length as files. destinations must not already exist.
...	Additional arguments, perhaps used by a filter function.
filter	A simple function taking as it's first argument a ShortReadQ instance and returning a modified ShortReadQ instance (e.g., with records or nucleotides removed), or a FilterRules instance specifying which records are to be removed.
compress	A logical(1) indicating whether the file should be gz-compressed. The default is TRUE.
yieldSize	Number of fastq records processed in each call to filter; increase this for (marginally) more efficient I/O at the expense of increased memory use.

Author(s)

Martin Morgan mtmorgan@fhcrc.org

Examples

```
## path to a convenient fastq file
sp <- SolexaPath(system.file('extdata', package='ShortRead'))
fl <- file.path(analysisPath(sp), "s_1_sequence.txt")

## filter reads to keep those with GC < 0.7
fun <- function(x) {
    gc <- alphabetFrequency(sread(x), baseOnly=TRUE)[,c("G", "C")]
    x[rowSums(gc) / width(x) < .7]
}
filterFastq(fl, tempfile(), filter=fun)

## trimEnds,character-method uses filterFastq internally
trimEnds(fl, "V", destinations=tempfile())
```

Intensity-class *(Legacy) "Intensity", "IntensityInfo", and "IntensityMeasure" base classes for short read image intensities*

Description

The Intensity, IntensityMeasure, and IntensityInfo classes represent and manipulate image intensity measures. Instances from the class may also contain information about measurement errors, and additional information about the reads from which the intensities are derived.

Intensity, and IntensityMeasure, are virtual classes, and cannot be created directly. Classes derived from IntensityMeasure (e.g., ArrayIntensity) and Intensity (e.g., SolexaIntensity) are used to represent specific technologies.

Objects from the Class

`ArrayIntensity` objects can be created with calls of the form `ArrayIntensity(array(0, c(1,2,3)))`.

Objects of derived classes can be created from calls such as the `SolexaIntensity` constructor, or more typically by parsing appropriate files (e.g., `readIntensities`).

Slots

Class `Intensity` has slots:

`readInfo`**:** Object of class `"IntensityInfo"` containing columns for the lane, tile, x, and y coordinates of the read.

`intensity`**:** Object of class `"IntensityMeasure"` containing image intensity data for each read and cycle.

`measurementError`**:** Object of class `"IntensityMeasure"` containing measures of image intensity uncertainty for each read and cycle.

`.hasMeasurementError`**:** Length 1 logical variable indicating whether intensity standard errors are included (internal use only).

Classes `IntensityInfo` and `IntensityMeasure` are virtual classes, and have no slots.

Extends

These classes extend `".ShortReadBase"`, directly.

Methods

Methods and accessor functions for `Intensity` include:

readIntensityInfo `signature(object = "Intensity")`: access the `readInfo` slot of `object`.

intensity `signature(object = "Intensity")`: access the `intensity` slot of `object`.

measurementError `signature(object = "Intensity")`: access the `nse` slot of `object`, or signal an error if no standard errors are available.

dim `signature(object = "Intensity")`: return the dimensions (e.g., number of reads by number of cycles) represented by `object`.

show `signature(object = "Intensity")`: provide a compact representation of the object.

Subsetting `"["` is available for the `IntensityMeasure` class; the `drop` argument to `"["` is ignored.

Subsetting with `"[["` is available for the `ArrayIntensity` class. The method accepts three arguments, corresponding to the read, base, and cycle(s) to be selected. The return value is the array (i.e., underlying data values) corresponding to the selected indices.

Author(s)

Martin Morgan <mtmorgan@fhcrc.org>

See Also

`readIntensities`

Examples

```
showMethods(class="Intensity", where=getNamespace("ShortRead"))
example(readIntensities)
```

MAQMapQA-class *(Legacy) Quality assessment summaries from MAQ map files*

Description

This class contains a list-like structure with summary descriptions derived from visiting one or more MAQMap files.

Objects from the Class

Objects of the class are usually produced by a qa method.

Slots

`.srlist`: Object of class `"list"`, containing data frames or lists of data frames summarizing the results of qa.

Extends

Class `"SRList"`, directly. Class `".QA"`, directly. Class `".SRUtil"`, by class "SRList", distance 2. Class `".ShortReadBase"`, by class ".QA", distance 2.

Methods

Accessor methods are inherited from the SRList class.

report `signature(x="MAQMapQA", ..., dest=tempfile(), type="html")`: produces an html file summarizing the QA results.

Author(s)

Martin Morgan <mtmorgan@fhcrc.org>

See Also

qa.

Examples

```
showClass("MAQMapQA")
```

qa *Perform quality assessment on short reads*

Description

This function is a common interface to quality assessment functions available in ShortRead. Results from this function may be displayed in brief, or integrated into reports using, e.g., report.

Usage

```
qa(dirPath, ...)
## S4 method for signature 'character'
qa(dirPath, pattern=character(0),
    type=c("fastq", "SolexaExport", "SolexaRealign", "Bowtie",
           "MAQMap", "MAQMapShort"),
    ...)
## S4 method for signature 'list'
qa(dirPath, ...)
```

Arguments

dirPath	A character vector or other object (e.g., SolexaPath; see showMethods, below) locating the data for which quality assessment is to be performed. See help pages for defined methods (by evaluating the example code, below) for details of available methods.
pattern	A character vector limiting the files in dirPath to be processed, as with list.files. Care should be taken to specify pattern to avoid reading unintended files.
type	The type of file being parsed; must be a character vector of length 1, selected from one of the types enumerated in the parameter.
...	Additional arguments used by methods.

sample=TRUE: Logical(1) indicating whether QA should be performed on a sample (default size 1000000) drawn from each FASTQ file, or from the entire file.

n: The number of reads to sample when processing FASTQ files.

Lpattern, Rpattern: A character vector or XString object to be matched to the left end of a sequence. If either Lpattern or Rpattern are provided, trimLRPatterns is invoked to produce a measure of adapter contamination. Mismatch rates are 0.1 on the left and 0.2 on the right, with a minimum overlap of 10 nt.

BPPARAM: How parallel evalutation will be performed. see BiocParallelParam; the default is BiocParallel::registered()[1].

Details

The most common use of this function provides a directory path and pattern identifying FASTQ files for quality assessment. The default is then to create a quality assessment report based on a random sample of n=1000000 reads from each file.

The following methods are defined, in addition to those on S4 formal classes documented elsewhere:

`qa,character-method` Quality assessment is performed on all files in directory `dirPath` whose file name matches `pattern`. The type of analysis performed is based on the `type` argument. Use `SolexaExport` when all files matching `pattern` are Solexa `_export.txt` files. Use `SolexaRealign` for Solexa `_realign.txt` files. Use `Bowtie` for Bowtie files. Use `MAQMapShort` for MAQ `map` files produced by MAQ versions below 0.70 and `MAQMap` for more recent output. Use `fastq` for collections of fastq-format files. Quality assessment details vary depending on data source.

`qa,list-method` `dirPath` is a list of objects, all of the same class and typically derived from `ShortReadQ`, on which quality assessment is performed. All elements of the list must have names, and these should be unique.

Value

An object derived from class `.QA`. Values contained in this object are meant for use by `report`

Author(s)

Martin Morgan <mtmorgan@fhcrc.org>

See Also

`.QA`, `SolexaExportQA` `MAQMapQA` `FastqQA`

Examples

```
dirPath <- system.file(package="ShortRead", "extdata", "E-MTAB-1147")
## sample 1M reads / file
qa <- qa(dirPath, "fastq.gz", BPPARAM=SerialParam())
if (interactive())
    browseURL(report(qa))

showMethods("qa", where=getNamespace("ShortRead"))
```

QA-class *(Updated) classes for representing quality assessment results*

Description

Classes derived from `.QA-class` represent results of quality assurance analyses.

Objects from the Class

Users create instances of many of these classes by calling the corresponding constructors, as documented on the help page for qa2. Classes constructed in this way include QACollate, QAFastqSource, QAAdapterContamination, QAFrequentSequence, QANucleotideByCycle, QANucleotideUse, QAQualityByCycle, QAQualityUse, QAReadQuality, and QASequenceUse.

The classes QASource, QAFiltered, QAFlagged and QASummary are generated internally, not by users.

Extends

.QA2 extends class ".ShortReadBase", directly.

QASummary is a virtual class extending .QA2; all user-creatable classes extend QASummary.

QASource extends QASummary. All classes used to represent raw data input (QAFastqSource) extend QASource.

QAData is a reference class, used to contain a single instance of the fastq used in all QA Summary steps.

QACollate extends .QA2. It contains a SimpleList instance with zero or more QASummary elements.

QA extends .QA2, and contains a SimpleList of zero or more QASummary elements. This class represents the results of the qa2 analysis.

Methods

Methods defined on this class include:

qa2 signature(object="QACollate", state, ...,verbose=FALSE) creates a QA report from the elements of QACollate. Methods on qa2 for objects extending class QASummary summarize QA statistics for that class, e.g., qa2,QAFrequentSequences-method implements the calculations required to summarize frequently used sequences, using data in state.

report signature(x="QA", ...) creates an HTML report. Methods on report for objects extending class QASummary are responsible for creating the html snippet for that QA component.

flag signature(object=".QA2", ..., verbose=FALSE) implements criteria to flag individual lanes as failing quality assessment. NOTE: flag is not fully implemented.

rbind signature(...="QASummary"): rbind multiple summary elements of the same class, as when these have been created by separately calculating statistics on a number of fastq files.

show signature(object = "SolexaExportQA"): Display an overview of the object contents.

Author(s)

Martin Morgan <mtmmorgan@fhcrc.org>

See Also

Specific classes derived from .QA2

Examples

```
getClass(".QA2", where=getNamespace("ShortRead"))
```

---

qa2 *(Updated) quality assessment reports on short reads*

Description

This page summarizes an updated approach to quality assessment reports in ShortRead.

Usage

```
## Input source for short reads
QAFastqSource(con = character(), n = 1e+06, readerBlockSize = 1e+08,
    flagNSequencesRange = NA_integer_, ...,
    html = system.file("template", "QASources.html", package="ShortRead"))
QAData(seq = ShortReadQ(), filter = logical(length(seq)), ...)

## Possible QA elements
QAFrequentSequence(useFilter = TRUE, addFilter = TRUE,
    n = NA_integer_, a = NA_integer_, flagK=.8, reportSequences = FALSE,
    ...)
QANucleotideByCycle(useFilter = TRUE, addFilter = TRUE, ...)
QANucleotideUse(useFilter = TRUE, addFilter = TRUE, ...)
QAQualityByCycle(useFilter = TRUE, addFilter = TRUE, ...)
QAQualityUse(useFilter = TRUE, addFilter = TRUE, ...)
QAReadQuality(useFilter = TRUE, addFilter = TRUE,
    flagK = 0.2, flagA = 30L, ...)
QASequenceUse(useFilter = TRUE, addFilter = TRUE, ...)
QAAdapterContamination(useFilter = TRUE, addFilter = TRUE,
    Lpattern = NA_character_, Rpattern = NA_character_,
    max.Lmismatch = 0.1, max.Rmismatch = 0.2, min.trim = 9L, ...)

## Order QA report elements
QACollate(src, ...)

## perform analysis
qa2(object, state, ..., verbose=FALSE)

## Outputs from qa2
QA(src, filtered, flagged, ...)
QAFiltered(useFilter = TRUE, addFilter = TRUE, ...)
QAFlagged(useFilter = TRUE, addFilter = TRUE, ...)

## Summarize results as html report
## S4 method for signature 'QA'
```

```
report(x, ..., dest = tempfile(), type = "html")

## additional methods; 'flag' is not fully implemented
flag(object, ..., verbose=FALSE)

## S4 method for signature 'QASummary'
rbind(..., deparse.level = 1)
```

Arguments

con	character(1) file location of fastq input, as used by FastqSampler.
n	integer(1) number of records to input, as used by FastqStreamer (QAFastqSource). integer(1) number of sequences to tag as 'frequent' (QAFrequentSequence).
readerBlockSize	integer(1) number of bytes to input, as used by FastqStreamer.
flagNSequencesRange	integer(2) minimum and maximum reads above which source files will be flagged as outliers.
html	character(1) location of the HTML template for summarizing this report element.
seq	ShortReadQ representation of fastq data.
filter	logical() vector with length equal to seq, indicating whether elements of seq are filtered (TRUE) or not.
useFilter, addFilter	logical(1) indicating whether the QA element should be calculating using the filtered (useFilter=TRUE) or all reads, and whether reads failing the QA element should be added to the filter used by subsequent steps (addFilter = TRUE) or not.
a	integer(1) count of number of sequences above which a read will be considered 'frequent' (QAFrequentSequence).
flagK, flagA	flagK numeric(1) between 0 and 1 indicating the fraction of frequent sequences greater than or equal to n or a above which a fastq file will be flagged (QAFrequentSequence). flagK numeric{1} between 0 and 1 and flagA integer(1) indicating that a run should be flagged when the fraction of reads with quality greater than or equal to flagA falls below threshold flagK.
reportSequences	logical(1) indicating whether frequent sequences are to be reported.
Lpattern, Rpattern, max.Lmismatch, max.Rmismatch, min.trim	Parameters influencing adapter identification, see matchPattern.
src	The source, e.g., QAFastqSource, on which the quality assessment report will be based.
object	An instance of class derived from QA on which quality metrics will be derived; for end users, this is usually the result of QACollate.
.	

state	The data on which quality assessment will be performed; this is not usually necessary for end-users.
verbose	logical(1) indicating whether progress reports should be reported.
filtered, flagged	Primarily for internal use, instances of QAFiltered and QAFlagged.
x	An instance of QA on which a report is to be generated.
dest	character(1) providing the directory in which the report is to be generated.
type	character(1) indicating the type of report to be generated; only "html" is supported.
deparse.level	see rbind.
...	Additional arguments, e.g., html to specify the location of the html source to use as a template for the report.

Details

Use QACollate to specify an order in which components of a QA report are to be assembled. The first argument is the data source (e.g., QAFastqSource).

Functions related to data input include:

QAFastqSource defines the location of fastq files to be included in the report. con is used to construct a FastqSampler instance, and records are processed using qa2,QAFastqSource-method.

QAData is a class for representing the data during the QA report generation pass; it is primarily for internal use.

Possible elements in a QA report are:

QAFrequentSequence identifies the most-commonly occuring sequences. One of n or a can be non-NA, and determine the number of frequent sequences reported. n specifies the number of most-frequent sequences to filter, e.g., n=10 would filter the top 10 most commonly occurring sequences; a provides a threshold frequency (count) above which reads are filtered. The sample is flagged when a fraction flagK of the reads are filtered.

reportSequences determines whether the most commonly occuring sequences, as determined by n or a, are printed in the html report.

QANucleotideByCycle reports nucleotide frequency as a function of cycle.

QAQualityByCycle reports average quality score as a function of cycle.

QAQualityUse summarizes overall nucleotide qualities.

QAReadQuality summarizes the distribution of read qualities.

QASequenceUse summarizes the cumulative distribution of reads occurring 1, 2, ... times.

QAAdapterContamination reports the occurrence of 'adapter' sequences on the left and / or right end of each read.

Value

An object derived from class .QA. Values contained in this object are meant for use by report

Author(s)

Martin Morgan <mtmorgan@fhcrc.org>

See Also

QA.

Examples

```
dirPath <- system.file(package="ShortRead", "extdata", "E-MTAB-1147")
fls <- dir(dirPath, "fastq.gz", full=TRUE)

coll <- QACollate(QAFastqSource(fls), QAReadQuality(),
    QAAdapterContamination(), QANucleotideUse(),
    QAQualityUse(), QASequenceUse(),
    QAFrequentSequence(n=10), QANucleotideByCycle(),
    QAQualityByCycle())
x <- qa2(coll, BPPARAM=SerialParam(), verbose=TRUE)

res <- report(x)
if (interactive())
    browseURL(res)
```

QualityScore *Construct objects indicating read or alignment quality*

Description

Use these functions to construct quality indicators for reads or alignments. See QualityScore for details of object content and methods available for manipulating them.

Usage

```
NumericQuality(quality = numeric(0))
IntegerQuality(quality = integer(0))
MatrixQuality(quality = new("matrix"))
FastqQuality(quality, ...)
SFastqQuality(quality, ...)
```

Arguments

quality	An object used to initialize the data structure. Appropriate objects are indicated in the constructors above for Numeric, Integer, and Matrix qualities. For FastqQuality and SFastqQuality, methods are defined for BStringSet, character, and missing.
...	Additional arguments, currently unused.

Value

Constructors return objects of the corresponding class derived from QualityScore.

Author(s)

Martin Morgan <mtmorgan@fhcrc.org>

See Also

QualityScore, readFastq, readAligned

Examples

```
nq <- NumericQuality(rnorm(20))
nq
quality(nq)
quality(nq[10:1])
```

QualityScore-class *Quality scores for short reads and their alignments*

Description

This class hierarchy represents quality scores for short reads. QualityScore is a virtual base class, with derived classes offering different ways of representing qualities. Methods defined on QualityScore are implemented in all derived classes.

Objects from the Class

Objects from the class are created using constructors (e.g., NumericQuality) named after the class name.

Defined classes are as follows:

QualityScore Virtual base class; instances cannot be instantiated.

NumericQuality A single numeric vector, where values represent quality scores on an arbitrary scale.

IntegerQuality A integer numeric vector, where values represent quality scores on an arbitrary scale.

MatrixQuality A rectangular matrix of quality scores, with rows representing reads and columns cycles. The content and interpretation of row and column entries is arbitrary; the rectangular nature implies quality scores from equal-length reads.

FastqQuality 'fastq' encoded quality scores stored in a BStringSet instance. Base qualities of a single read are represented as an ASCII character string. The integer-valued quality score of a single base is encoded as its ASCII equivalent plus 33. The precise definition of the integer-valued quality score is unspecified, but is usually a Phred score; the meaning can be determined from the source of the quality scores. Multiple reads are stored as a BStringSet, and so can be of varying lengths.

SolexaQuality As with `FastqQuality`, but with integer qualities encoded as ASCII equivalent plus 64.

Extends

Class `".ShortReadBase"`, directly.

Methods

The following methods are defined on all `QualityScore` and derived classes:

[`signature(x = "QualityScore", i = "ANY", j = "missing")`

[`signature(x = "MatrixQuality", i = "ANY", j = "missing")`:

Subset the object, with index `i` indicating the reads for which quality scores are to be extracted. The class of the result is the same as the class of `x`. It is an error to provide any argument other than `i`.

[[`signature(x = "QualityScore", i = "ANY", j = "ANY")`:

Subset the object, returning the quality score (e.g., numeric value) of the `i`th read.

[[`signature(x = "MatrixQuality", i = "ANY", j = "ANY")`:

Returns the vector of quality scores associated with the `i`th read.

dim `signature(x = "MatrixQuality")`:

The integer(2) dimension (e.g., number of reads, read width) represented by the quality score.

length `signature(x = "QualityScore")`:

length `signature(x = "MatrixQuality")`:

The integer(1) length (e.g., number of reads) represented by the quality score. Note that `length` of `MatrixQuailty` is the number of rows of the corresponding matrix, and not the length of the corresponding numeric vector.

append `signature(x = "QualityScore", values = "QualityScore")`: append `values` after `x`.

width `signature(x = "QualityScore")`:

width `signature(x = "NumericQuality")`:

width `signature(x = "MatrixQuality")`:

width `signature(x = "FastqQuality")`:

A numeric vector with length equal to the number of quality scores, and value equal to the number of quality scores for each read. For instance, a `FastqQuality` will have widths equal to the number of nucleotides in the underlying short read.

show `signature(object = "QualityScore")`:

show `signature(object = "NumericQuality")`:

show `signature(object = "FastqQuality")`:

provide a brief summary of the object content.

detail `signature(x = "QualityScore")`:

provide a more detailed view of object content.

The following methods are defined on specific classes:

alphabet `signature(x = "FastqQuality", ...)`: Return a character vector of valid quality characters.

encoding `signature(x = "FastqQuality", ...)`, `signature(x = "SFastqQuality", ...)`: Returns a named character vector of integer encodings.

alphabetFrequency `signature(stringSet = "FastqQuality")`:

Apply `alphabetFrequency` to quality scores, returning a matrix as described in `alphabetFrequency`.

alphabetByCycle `signature(stringSet = "FastqQuality")`:

Apply `alphabetByCycle` to quality scores, returning a matrix as described in `alphabetByCycle`.

alphabetScore `signature(object = "FastqQuality")`:

alphabetScore `signature(object = "SFastqQuality")`:

alphabetScore `signature(object = "PhredQuality")`:

Apply `alphabetScore` (i.e., summed base quality, per read) to `object`.

coerce `signature(from = "FastqQuality", to = "numeric")`:

coerce `signature(from = "FastqQuality", to = "matrix")`:

coerce `signature(from = "FastqQuality", to = "PhredQuality")`:

coerce `signature(from = "SFastqQuality", to = "matrix")`:

coerce `signature(from = "SFastqQuality", to = "SolexaQuality")`:

Use `as(from, "matrix")`) and similar to coerce objects of class `from` to class `to`, using the quality encoding implied by the class. When `to` is "matrix", the result is a matrix of type `integer` with number of columns equal to the maximum width of `from`; elements `i, j` with `j > width(from)[i]` have value `NA_integer_`. The result always represents the integer encoding of the corresponding quality string.

reverse `signature(x = "FastqQuality", ...)`: reverse the quality sequence.

narrow `signature(x = "FastqQuality", start = NA, end = NA, width = NA, use.names = TRUE)`: 'narrow' `quality` so that scores are between `start` and `end` bases, according to `narrow` in the `IRanges` package.

trimTailw `signature(object="FastqQuality", k="integer",a="character", halfwidth="integer", ..., ranges=FALSE)`: trim trailing nucleotides when a window of width 2 * halfwidth + 1 contains `k` or more quality scores falling at or below `a`.

trimTails `signature(object="FastqQuality", k="integer",a="character", successive=FALSE, ..., ranges=FALSE)`: trim trailing scores if `k` scores fall below the quality encoded by `a`. If `successive=FALSE`, the k'th failing score and all subseqent scores are trimmed. If `successive=TRUE`, failing scores must occur successively; the sequence is trimmed from the first of the successive failing score.

srorder `signature(x = "FastqQuality")`:

srrank `signature(x = "FastqQuality")`:

srduplicated `signature(x = "FastqQuality")`:

Apply `srsort`, `srorder`, `srrank`, and `srduplicated` to quality scores, returning objects as described on the appropriate help page.

Integer representations of `SFastqQuality` and `FastqQuality` can be obtained with `as(x, "matrix")`.

Author(s)

Martin Morgan <mtmorgan@fhcrc.org>

See Also

NumericQuality and other constructors.

Examples

```
names(slot(getClass("QualityScore"), "subclasses"))
encoding(FastqQuality())
encoding(SFastqQuality())
```

---

readAligned *(Legacy) Read aligned reads and their quality scores into R representations*

Description

Import files containing aligned reads into an internal representation of the alignments, sequences, and quality scores. Most methods (see 'details' for exceptions) read all files into a single R object.

Usage

```
readAligned(dirPath, pattern=character(0), ...)
```

Arguments

dirPath	A character vector (or other object; see methods defined on this generic) giving the directory path (relative or absolute; some methods also accept a character vector of file names) of aligned read files to be input.
pattern	The (grep-style) pattern describing file names to be read. The default (character(0)) results in (attempted) input of all files in the directory.
...	Additional arguments, used by methods. When dirPath is a character vector, the argument type must be provided. Possible values for type and their meaning are described below. Most methods implement filter=srFilter(), allowing objects of SRFilter to selectively returns aligned reads.

Details

There is no standard aligned read file format; methods parse particular file types.

The readAligned,character-method interprets file types based on an additional type argument. Supported types are:

`type="SolexaExport"` This type parses `.*_export.txt` files following the documentation in the Solexa Genome Alignment software manual, version 0.3.0. These files consist of the following columns; consult Solexa documentation for precise descriptions. If parsed, values can be retrieved from `AlignedRead` as follows:

Machine see below
Run number stored in `alignData`
Lane stored in `alignData`
Tile stored in `alignData`
X stored in `alignData`
Y stored in `alignData`
Multiplex index see below
Paired read number see below
Read `sread`
Quality `quality`
Match chromosome `chromosome`
Match contig `alignData`
Match position `position`
Match strand `strand`
Match description Ignored
Single-read alignment score `alignQuality`
Paired-read alignment score Ignored
Partner chromosome Ignored
Partner contig Ignored
Partner offset Ignored
Partner strand Ignored
Filtering `alignData`

The following optional arguments, set to `FALSE` by default, influence data input

withMultiplexIndex When `TRUE`, include the multiplex index as a column `multiplexIndex` in `alignData`.

withPairedReadNumber When `TRUE`, include the paired read number as a column `pairedReadNumber` in `alignData`.

withId When `TRUE`, construct an identifier string as 'Machine_Run:Lane:Tile:X:Y#multiplexIndex/pairedReadNumber The substrings '#multiplexIndex' and '/pairedReadNumber' are not present if `withMultiplexIndex=FALSE` or `withPairedReadNumber=FALSE`.

withAll A convencience which, when `TRUE`, sets all `with*` values to `TRUE`.

Note that not all paired read columns are interpreted. Different interfaces to reading alignment files are described in `SolexaPath` and `SolexaSet`.

`type="SolexaPrealign"` See SolexaRealign

`type="SolexaAlign"` See SolexaRealign

`type="SolexaRealign"` These types parse `s_L_TTTT_prealign.txt`, `s_L_TTTT_align.txt` or `s_L_TTTT_realign.txt` files produced by default and eland analyses. From the Solexa documentation, `align` corresponds to unfiltered first-pass alignments, `prealign` adjusts alignments for error rates (when available), `realign` filters alignments to exclude clusters failing to pass quality criteria.

Because base quality scores are not stored with alignments, the object returned by `readAligned` scores all base qualities as `-32`.

If parsed, values can be retrieved from `AlignedRead` as follows:

Sequence stored in `sread`

Best score stored in `alignQuality`

Number of hits stored in `alignData`

Target position stored in `position`

Strand stored in `strand`

Target sequence Ignored; parse using `readXStringColumns`

Next best score stored in `alignData`

`type="SolexaResult"` This parses `s_L_eland_results.txt` files, an intermediate format that does not contain read or alignment quality scores.

Because base quality scores are not stored with alignments, the object returned by `readAligned` scores all base qualities as `-32`.

Columns of this file type can be retrieved from `AlignedRead` as follows (description of columns is from Table 19, Genome Analyzer Pipeline Software User Guide, Revision A, January 2008):

Id Not parsed

Sequence stored in `sread`

Type of match code Stored in `alignData` as `matchCode`. Codes are (from the Eland manual): NM (no match); QC (no match due to quality control failure); RM (no match due to repeat masking); U0 (best match was unique and exact); U1 (best match was unique, with 1 mismatch); U2 (best match was unique, with 2 mismatches); R0 (multiple exact matches found); R1 (multiple 1 mismatch matches found, no exact matches); R2 (multiple 2 mismatch matches found, no exact or 1-mismatch matches).

Number of exact matches stored in `alignData` as `nExactMatch`

Number of 1-error mismatches stored in `alignData` as `nOneMismatch`

Number of 2-error mismatches stored in `alignData` as `nTwoMismatch`

Genome file of match stored in `chromosome`

Position stored in `position`

Strand (direction of match) stored in `strand`

'N' treatment stored in `alignData`, as `NCharacterTreatment`. '.' indicates treatment of 'N' was not applicable; 'D' indicates treatment as deletion; 'I' indicates treatment as insertion

Substitution error stored in `alignData` as `mismatchDetailOne` and `mismatchDetailTwo`. Present only for unique inexact matches at one or two positions. Position and type of first substitution error, e.g., 11A represents 11 matches with 12th base an A in reference but not read. The reference manual cited below lists only one field (`mismatchDetailOne`), but two are present in files seen in the wild.

`type="MAQMap", records=-1L` Parse binary `map` files produced by MAQ. See details in the next section. The `records` option determines how many lines are read; `-1L` (the default) means that all records are input. For `type="MAQMap"`, `dir` and `pattern` must match a single file.

`type="MAQMapShort", records=-1L` The same as `type="MAQMap"` but for map files made with Maq prior to version 0.7.0. (These files use a different maximum read length [64 instead of 128], and are hence incompatible with newer Maq map files.). For `type="MAQMapShort"`, `dir` and `pattern` must match a single file.

`type="MAQMapview"` Parse alignment files created by MAQ's 'mapiew' command. Interpretation of columns is based on the description in the MAQ manual, specifically

> ...each line consists of read name, chromosome, position, strand, insert size from the outer coordinates of a pair, paired flag, mapping quality, single-end mapping quality, alternative mapping quality, number of mismatches of the best hit, sum of qualities of mismatched bases of the best hit, number of 0-mismatch hits of the first 24bp, number of 1-mismatch hits of the first 24bp on the reference, length of the read, read sequence and its quality.

The read name, read sequence, and quality are read as `XStringSet` objects. Chromosome and strand are read as `factors`. Position is `numeric`, while mapping quality is `numeric`. These fields are mapped to their corresponding representation in `AlignedRead` objects.

Number of mismatches of the best hit, sum of qualities of mismatched bases of the best hit, number of 0-mismatch hits of the first 24bp, number of 1-mismatch hits of the first 24bp are represented in the `AlignedRead` object as components of `alignData`.

Remaining fields are currently ignored.

`type="Bowtie"` Parse alignment files created with the Bowtie alignment algorithm. Parsed columns can be retrieved from `AlignedRead` as follows:

Identifier `id`

Strand `strand`

Chromosome `chromosome`

Position `position`; see comment below

Read `sread`; see comment below

Read quality `quality`; see comments below

Similar alignments `alignData`, 'similar' column; Bowtie v. 0.9.9.3 (12 May, 2009) documents this as the number of other instances where the same read aligns against the same reference characters as were aligned against in this alignment. Previous versions marked this as 'Reserved'

Alignment mismatch locations `alignData` 'mismatch', column

NOTE: the default quality encoding changes to `FastqQuality` with **ShortRead** version 1.3.24.

This method includes the argument `qualityType` to specify how quality scores are encoded. Bowtie quality scores are 'Phred'-like by default, with `qualityType='FastqQuality'`, but can be specified as 'Solexa'-like, with `qualityType='SFastqQuality'`.

Bowtie outputs positions that are 0-offset from the left-most end of the `+` strand. `ShortRead` parses position information to be 1-offset from the left-most end of the `+` strand.

Bowtie outputs reads aligned to the `-` strand as their reverse complement, and reverses the quality score string of these reads. `ShortRead` parses these to their original sequence and orientation.

`type="SOAP"` Parse alignment files created with the SOAP alignment algorithm. Parsed columns can be retrieved from `AlignedRead` as follows:

id `id`

seq `sread`; see comment below

qual quality; see comment below

number of hits alignData

a/b alignData (pairedEnd)

length alignData (alignedLength)

+/- strand

chr chromosome

location position; see comment below

types alignData (typeOfHit: integer portion; hitDetail: text portion)

This method includes the argument qualityType to specify how quality scores are encoded. It is unclear from SOAP documentation what the quality score is; the default is 'Solexa'-like, with qualityType='SFastqQuality', but can be specified as 'Phred'-like, with qualityType='FastqQuality'.

SOAP outputs positions that are 1-offset from the left-most end of the + strand. ShortRead preserves this representation.

SOAP reads aligned to the - strand are reported by SOAP as their reverse complement, with the quality string of these reads reversed. ShortRead parses these to their original sequence and orientation.

Value

A single R object (e.g., AlignedRead) containing alignments, sequences and qualities of all files in dirPath matching pattern. There is no guarantee of order in which files are read.

Author(s)

Martin Morgan <mtmorgan@fhcrc.org>, Simon Anders <anders@ebi.ac.uk> (MAQ map)

See Also

The AlignedRead class.

Genome Analyzer Pipeline Software User Guide, Revision A, January 2008.

The MAQ reference manual, http://maq.sourceforge.net/maq-manpage.shtml#5, 3 May, 2008.

The Bowtie reference manual, http://bowtie-bio.sourceforge.net, 28 October, 2008.

The SOAP reference manual, http://soap.genomics.org.cn/soap1, 16 December, 2008.

Examples

```
sp <- SolexaPath(system.file("extdata", package="ShortRead"))
ap <- analysisPath(sp)
## ELAND_EXTENDED
(aln0 <- readAligned(ap, "s_2_export.txt", "SolexaExport"))
## PhageAlign
(aln1 <- readAligned(ap, "s_5_.*_realign.txt", "SolexaRealign"))

## MAQ
dirPath <- system.file('extdata', 'maq', package='ShortRead')
list.files(dirPath)
## First line
```

```
readLines(list.files(dirPath, full.names=TRUE)[[1]], 1)
countLines(dirPath)
## two files collapse into one
(aln2 <- readAligned(dirPath, type="MAQMapview"))

## select only chr1-5.fa, '+' strand
filt <- compose(chromosomeFilter("chr[1-5].fa"),
                strandFilter("+"))
(aln3 <- readAligned(sp, "s_2_export.txt", filter=filt))
```

`readBaseQuality` *(Legacy) Read short reads and their quality scores into R representations*

Description

`readBaseQuality` reads all base call files in a directory `dirPath` whose file name matches `seqPattern` and all quality score files whose name matches `prbPattern`, returning a compact internal representation of the sequences, and quality scores in the files. Methods read all files into a single R object.

Usage

```
readBaseQuality(dirPath, ...)
## S4 method for signature 'character'
readBaseQuality(dirPath, seqPattern=character(0),
prbPattern=character(0), type=c("Solexa"), ...)
```

Arguments

`dirPath`	A character vector (or other object; see methods defined on this generic) giving the directory path (relative or absolute) of files to be input.
`seqPattern`	The (`grep`-style) pattern describing base call file names to be read. The default (`character(0)`) results in (attempted) input of all files in the directory.
`prbPattern`	The (`grep`-style) pattern describing quality score file names to be read. The default (`character(0)`) results in (attempted) input of all files in the directory.
`type`	The type of file to be parsed. Supported types include: `Solexa`: parse reads and their qualities from `_seq.txt` and `_prb.txt`-formatted files, respectively.
`...`	Additional arguments, perhaps used by methods.

Value

A single R object (e.g., `ShortReadQ`) containing sequences and qualities of all files in `dirPath` matching `seqPattern` and `prbPattern` respectively. There is no guarantee of order in which files are read.

Author(s)

Patrick Aboyoun <paboyoun@fhcrc.org>

See Also

A ShortReadQ object.

readXStringColumns, readPrb

Examples

```
sp <- SolexaPath(system.file("extdata", package="ShortRead"))
readBaseQuality(sp, seqPattern="s_1.*_seq.txt", prbPattern="s_1.*_prb.txt")
```

---

readBfaToc *(Legacy) Get a list of the sequences in a Maq .bfa file*

Description

As coverage needs to know the lengths of the reference sequences, this function is provided which extracts this information from a .bfa file (Maq's "binary FASTA" format).

Usage

```
readBfaToc( bfafile )
```

Arguments

bfafile	The file name of the .bfa file.

Value

An integer vector with one element per reference sequence found in the .bfa file, each vector element named with the sequence name and having the sequence length as value.

Author(s)

Simon Anders, EMBL-EBI, <sanders@fs.tum.de>

(Note: The C code for this function incorporates code from Li Heng's MAQ software, (c) Li Heng and released by him under GPL 2.

readFasta *Read and write FASTA files to or from ShortRead objects*

Description

readFasta reads all FASTA-formated files in a directory dirPath whose file name matches pattern pattern, returning a compact internal representation of the sequences and quality scores in the files. Methods read all files into a single R object; a typical use is to restrict input to a single FASTA file.

writeFasta writes an object to a single file, using mode="w" (the default) to create a new file or mode="a" append to an existing file. Attempting to write to an existing file with mode="w" results in an error.

Usage

```
readFasta(dirPath, pattern = character(0), ...,
    nrec=-1L, skip=0L)
## S4 method for signature 'character'
readFasta(dirPath, pattern = character(0), ...,
    nrec=-1L, skip=0L)
writeFasta(object, file, mode="w", ...)
## S4 method for signature 'DNAStringSet'
writeFasta(object, file, mode="w", ...)
```

Arguments

dirPath	A character vector giving the directory path (relative or absolute) or single file name of FASTA files to be read.
pattern	The (grep-style) pattern describing file names to be read. The default (character(0)) results in (attempted) input of all files in the directory.
object	An object to be output in fasta format.
file	A length 1 character vector providing a path to a file to the object is to be written to.
mode	A length 1 character vector equal to either 'w' or 'a' to write to a new file or append to an existing file, respectively.
...	Additional arguments used by methods or, for writeFasta, writeXStringSet.
nrec	See ?readDNAStringSet.
skip	See ?readDNAStringSet.

Value

readFasta returns a DNAStringSet. containing sequences and qualities contained in all files in dirPath matching pattern. There is no guarantee of order in which files are read.

writeFasta is invoked primarily for its side effect, creating or appending to file file. The function returns, invisibly, the length of object, and hence the number of records written. There is a writeFasta method for any class derived from ShortRead.

Author(s)

Martin Morgan

Examples

```
showMethods("readFasta")

showMethods("writeFasta")

f1 <- system.file("extdata", "someORF.fa", package="Biostrings")

rfa <- readFasta(f1)
sread(rfa)
id(rfa)

sp <- SolexaPath(system.file('extdata', package='ShortRead'))
rfq <- readFastq(analysisPath(sp), pattern="s_1_sequence.txt")

file <- tempfile()
writeFasta(rfq, file)
readLines(file, 8)

writeFasta(sread(rfq), file)  # no 'id's
```

---

readFastq *Read, write, and count records in FASTQ-formatted files*

Description

`readFastq` reads all FASTQ-formated files in a directory `dirPath` whose file name matches pattern `pattern`, returning a compact internal representation of the sequences and quality scores in the files. Methods read all files into a single R object; a typical use is to restrict input to a single FASTQ file.

`writeFastq` writes an object to a single `file`, using `mode="w"` (the default) to create a new file or `mode="a"` append to an existing file. Attempting to write to an existing file with `mode="w"` results in an error.

`countFastq` counts the nubmer of records, nucleotides, and base-level quality scores in one or several fastq files.

Usage

```
readFastq(dirPath, pattern=character(0), ...)
## S4 method for signature 'character'
readFastq(dirPath, pattern=character(0), ..., withIds=TRUE)

writeFastq(object, file, mode="w", full=FALSE, compress=TRUE, ...)

countFastq(dirPath, pattern=character(0), ...)
## S4 method for signature 'character'
countFastq(dirPath, pattern=character(0), ...)
```

Arguments

dirPath	A character vector (or other object; see methods defined on this generic) giving the directory path (relative or absolute) or single file name of FASTQ files to be read.
pattern	The (grep-style) pattern describing file names to be read. The default (character(0)) results in (attempted) input of all files in the directory.
object	An object to be output in fastq format. For methods, use showMethods(object, where=getNamespace("ShortRead")).
file	A length 1 character vector providing a path to a file to the object is to be written to.
mode	A length 1 character vector equal to either 'w' or 'a' to write to a new file or append to an existing file, respectively.
full	A logical(1) indicating whether the identifier line should be repeated full=TRUE or omitted full=FALSE on the third line of the fastq record.
compress	A logical(1) indicating whether the file should be gz-compressed. The default is TRUE.
...	Additional arguments. In particular, qualityType and filter:
	qualityType: Representation to be used for quality scores, must be one of Auto (choose Illumina base 64 encoding SFastqQuality if all characters are ASCII-encoded as greater than 58 : and some characters are greater than 74 J), FastqQuality (Phred-like base 33 encoding), SFastqQuality (Illumina base 64 encoding).
	filter: An object of class srFilter, used to filter objects of class ShortReadQ at input.
withIds	logical(1) indicating whether identifiers should be read from the fastq file.

Details

The fastq format is not quite precisely defined. The basic definition used here parses the following four lines as a single record:

```
@HWI-EAS88_1_1_1_1001_499
GGACTTTGTAGGATACCCTCGCTTTCCTTCTCCTGT
+HWI-EAS88_1_1_1_1001_499
]]]]]]]]]]]]Y]Y]]]]]]]]]]]]VCHVMPLAS
```

The first and third lines are identifiers preceded by a specific character (the identifiers are identical, in the case of Solexa). The second line is an upper-case sequence of nucleotides. The parser recognizes IUPAC-standard alphabet (hence ambiguous nucleotides), coercing . to - to represent missing values. The final line is an ASCII-encoded representation of quality scores, with one ASCII character per nucleotide.

The encoding implicit in Solexa-derived fastq files is that each character code corresponds to a score equal to the ASCII character value minus 64 (e.g., ASCII @ is decimal 64, and corresponds to a Solexa quality score of 0). This is different from BioPerl, for instance, which recovers quality

scores by subtracting 33 from the ASCII character value (so that, for instance, !, with decimal value 33, encodes value 0).

The BioPerl description of fastq asserts that the first character of line 4 is a !, but the current parser does not support this convention.

writeFastq creates files following the specification outlined above, using the IUPAC-standard alphabet (hence, sequences containing '.' when read will be represented by '-' when written).

Value

readFastq returns a single R object (e.g., ShortReadQ) containing sequences and qualities contained in all files in dirPath matching pattern. There is no guarantee of order in which files are read.

writeFastq is invoked primarily for its side effect, creating or appending to file file. The function returns, invisibly, the length of object, and hence the number of records written.

countFastq returns a data.frame with row names equal to the base (file) name of the fastq file, and columns records, nucleotides, and scores, corresponding to tally of each entity in each file. Parsing mistakes from poorly formmated files result in an error.

Author(s)

Martin Morgan

See Also

The IUPAC alphabet in Biostrings.

http://www.bioperl.org/wiki/FASTQ_sequence_format for the BioPerl definition of fastq.

Solexa documentation 'Data analysis - documentation : Pipeline output and visualisation'.

Examples

```
methods(readFastq)
methods(writeFastq)
methods(countFastq)

sp <- SolexaPath(system.file('extdata', package='ShortRead'))
rfq <- readFastq(analysisPath(sp), pattern="s_1_sequence.txt")
sread(rfq)
id(rfq)
quality(rfq)

## SolexaPath method 'knows' where FASTQ files are placed
rfq1 <- readFastq(sp, pattern="s_1_sequence.txt")
rfq1

file <- tempfile()
writeFastq(rfq, file)
readLines(file, 8)
countFastq(file)
```

readIntensities *(Legacy) Read Illumina image intensity files*

Description

readIntensities reads image 'intensity' files (such as Illumina's _int.txt and (optionally) _nse.txt) into a single object.

Usage

```
readIntensities(dirPath, pattern=character(0), ...)
```

Arguments

dirPath	Directory path or other object (e.g., SolexaPath) for which methods are defined.
pattern	A length 1 character vector representing a regular expression to be combined with dirPath, as described below, to match files to be summarized.
...	Additional arguments used by methods.

Details

Additional methods are defined on specific classes, see, e.g., SolexaPath.

The readIntensities,character-method contains an argument type that determines how intensities are parsed. Use the type argument to readIntensities,character-method, as described below. All readIntensities,character methods accepts the folling arguments:

withVariability: Include estimates of variability (i.e., from parsing _nse files).

verbose: Report on progress when starting to read each file.

The supported types and their signatures are:

type="RtaIntensity" Intensities are read from Illumina _cif.txt and _cnf.txt-style files. The signature for this method is

```
dirPath, pattern=character(0), ..., type="RtaIntensity", lane=integer(0), cycles=integer(0),
cycleIteration=1L, tiles=integer(0), laneName=sprintf("L cycleNames=sprintf("C
tileNames=sprintf("s_ posNames=sprintf("s_ withVariability=TRUE, verbose=FALSE
```

lane: integer(1) identifying the lane in which cycles and tiles are to be processed.

cycles: integer() enumerating cycles to be processed.

cycleIteration: integer(1) identifying the iteration of the base caller to be summarized

tiles: integer() enumerating tile numbers to be summarized.

laneName, cycleNames, tileNames, posNames: character() vectors identifying the lane and cycle directories, and the 'pos' and tile file names (excluding the '.cif' or '.cnf' extension) to be processed.

The dirPath and pattern arguments are combined as list.files(dirPath, pattern), and must identify a single directory. Most uses of this function will focus on a single tile (specified with, e.g., tiles=1L); the laneName, cycleNames, tileNames, and posNames parameters are designed to work with the default Illumina pipeline and do not normally need to be specified.

type="IparIntensity" Intensities are read from Solexa _pos.txt, _int.txt.p, _nse.txt.p-style file triplets. The signature for this method is

dirPath, pattern=character(0), ...,type="IparIntensity", intExtension="_int.txt.p.gz", nseExtension="_nse.txt.p.gz", posExtension="_pos.txt", withVariability=TRUE, verbose=FALSE

Files to be parsed are determined as, e.g., paste(pattern, intExtension, sep="").

type="SolexaIntensity" Intensities are read from Solexa _int.txt and _nse.txt-style files. The signature for this method is

dirPath, pattern=character(0), ...,type="SolexaIntensity", intExtension="_int.txt", nseExtension="_nse.txt", withVariability=TRUE, verbose=FALSE

Files to be parsed are determined as, e.g., paste(pattern, intExtension, sep="").

Value

An object derived from class Intensity.

Author(s)

Martin Morgan <mtmorgan@fhcrc.org>, Michael Muratet <mmuratet@hudsonalpha.org> (RTA).

Examples

```
fl <- system.file("extdata", package="ShortRead")
sp <- SolexaPath(fl)
int <- readIntensities(sp)
int
intensity(int)[1,,]   # one read
intensity(int)[[1:2,,]]         # two reads, as 'array'
head(rowMeans(intensity(int))) # treated as 'array'
head(pData(readIntensityInfo(int)))

## Not run: ## RTA Lane 2, cycles 1:80, cycle iteration 1, tile 3
int <- readIntensities("Data/Intensities", type="RtaIntensity",
                       lane=2, cycles=1:80, tiles=3)

## End(Not run)
```

---

readPrb *(Legacy) Read Solexa prb files as fastq-style quality scores*

Description

readPrb reads all _prb.txt files in a directory into a single object. Most methods (see details) do this by identifying the maximum base call quality for each cycle and read, and representing this as an ASCII-encoded character string.

Usage

```
readPrb(dirPath, pattern = character(0), ...)
```

Arguments

dirPath	Directory path or other object (e.g., SolexaPath for which methods are defined.
pattern	Regular expression matching names of _prb files to be summarized.
...	Additional arguments, unused.

Details

The readPrb,character-method contains an argument as that determines the value of the returned object, as follows.

as="SolexaEncoding" The ASCII encoding of the maximum per cycle and read quality score is encoded using Solexa conventions.

as="FastqEncoding" The ASCII encoding of the maximum per cycle and read quality score is encoded using Fastq conventions, i.e., ! has value 0.

as="IntegerEncoding" The maximum per cycle and read quality score is returned as a in integer value. Values are collated into a matrix with number of rows equal to number of reads, and number of columns equal to number of cycles.

as="array" The quality scores are *not* summarized; the return value is an integer array with dimensions corresponding to reads, nucleotides, and cycles.

Value

An object of class QualityScore, or an integer matrix.

Author(s)

Martin Morgan <mtmorgan@fhcrc.org>

Examples

```
fl <- system.file("extdata", package="ShortRead")
sp <- SolexaPath(fl)
readPrb(sp, "s_1.*_prb.txt") # all tiles to a single file
```

readQseq *(Legacy) Read Solexa qseq files as fastq-style quality scores*

Description

readQseq reads all files matching pattern in a directory into a single ShortReadQ-class object. Information on machine, lane, tile, x, and y coordinates, filtering status, and read number are not returned (although filtering status can be used to selectively include reads as described below).

Usage

```
readQseq(dirPath, pattern = character(0), ...,
         as=c("ShortReadQ", "DataFrame", "XDataFrame"),
         filtered=FALSE,
         verbose=FALSE)
```

Arguments

dirPath	Directory path or other object (e.g., SolexaPath) for which methods are defined.
pattern	Regular expression matching names of _qseq files to be summarized.
...	Additional argument, passed to I/O functions.
as	character(1) indicating the class of the return type. "XDataFrame" is included for backward compatibility, but is no longer supported.
filtered	logical(1) indicating whether to include only those reads passing Solexa filtering?
verbose	logical(1) indicating whether to report on progress during evaluation.

Value

An object of class ShortReadQ.

Author(s)

Martin Morgan <mtmorgan@fhcrc.org>

Examples

```
fl <- system.file("extdata", package="ShortRead")
sp <- SolexaPath(fl)
readQseq(sp)
```

readXStringColumns *Read one or more columns into XStringSet (e.g., DNAStringSet) objects*

Description

This function allows short read data components such as DNA sequence, quality scores, and read names to be read in to XStringSet (e.g., DNAStringSet, BStringSet) objects. One or several files of identical layout can be specified.

Usage

```
readXStringColumns(dirPath, pattern=character(0),
                   colClasses=list(NULL),
                   nrows=-1L, skip=0L,
                   sep = "\t", header = FALSE, comment.char="#")
```

Arguments

dirPath	A character vector giving the directory path (relative or absolute) of files to be read.
pattern	The (grep-style) pattern describing file names to be read. The default (character(0)) reads all files in dirPath. All files are expected to have identical numbers of columns.
colClasses	A list of length equal to the number of columns in a file. Columns with corresponding colClasses equal to NULL are ignored. Other entries in colClasses are expected to be character strings describing the base class for the XStringSet. For instance a column of DNA sequences would be specified as "DNAString". The column would be parsed into a DNAStringSet object.
nrows	A length 1 integer vector describing the maximum number of XString objects to read into the set. Reads may come from more than one file when dirPath and pattern parse several files and nrow is greater than the number of reads in the first file.
skip	A length 1 integer vector describing how many lines to skip at the start of each file.
sep	A length 1 character vector describing the column separator.
header	A length 1 logical vector indicating whether files include a header line identifying columns. If present, the header of the first file is used to name the returned values.
comment.char	A length 1 character vector, with a single character that, when appearing at the start of a line, indicates that the entire line should be ignored. Currently there is no way to use comment characters in other than the first position of a line.

Value

A list, with each element containing an `XStringSet` object of the type corresponding to the non-NULL elements of `colClasses`.

Author(s)

Martin Morgan <mtmorgan@fhcrc.org>

Examples

```
## valid character strings for colClasses
names(slot(getClass("XString"), "subclasses"))

dirPath <- system.file('extdata', 'maq', package='ShortRead')

colClasses <- rep(list(NULL), 16)
colClasses[c(1, 15, 16)] <- c("BString", "DNAString", "BString")

## read one file
readXStringColumns(dirPath, "out.aln.1.txt", colClasses=colClasses)

## read all files into a single object for each column
res <- readXStringColumns(dirPath, colClasses=colClasses)
```

---

renewable *Renew (update) a ShortRead object with new values*

Description

Use `renew` to update an object defined in **ShortRead** with new values. Discover update-able classes and values with `renewable`.

Usage

```
renewable(x, ...)
renew(x, ...)
```

Arguments

`x`	For `renewable`: `missing`, `character(1)`, or a class defined in the **ShortRead** package. For `renew`: an instance of a class defined in the **ShortRead** package.
`...`	For `renewable`, ignored. For `renew`, named arguments identifying which parts of `x` are to be renewed.

Details

When invoked with no arguments `renewable` returns a character vector naming classes that can be renewed.

When invoked with a `character(1)` or an instance of a **ShortRead** class, a list of the names and values of the elements that can be renewed. When `x` is a character vector naming a virtual class, then each element of the returned list is a non-virtual descendant of that class that can be used in renewal. This is not fully recursive.

`renew` is always invoked with the `x` argument being an instance of a class identified by `renewable()`. Remaining arguments are name-value pairs identifying the components of `x` that are to be renewed (updated). The name-value pairs must be consistent with `renewable(x)`. The resulting object is checked for validity. Multiple components of the object can be updated in a single call to `renew`, allowing comparatively efficient complex transformations.

Value

`renewable()` returns a character vector of renewable classes.

`renewable(x)` returns a named list. The names correspond to renewable classes, and the elements of the list correspond to renewable components of the class.

`renew(x, ...)` returns an object of the same class as `x`, but with components of `x` replaced by the named values of `...`.

Author(s)

Martin Morgan <mtmorgan@fhcrc.org>

Examples

```
## discovery
renewable()
renewable("AlignedRead")
renewable("QualityScore") ## instantiable classes

## example data
sp <- SolexaPath(system.file("extdata", package="ShortRead"))
ap <- analysisPath(sp)
filt <- chromosomeFilter("chr[[:digit:]+].fa")
aln <- readAligned(ap, "s_2_export.txt", "SolexaExport",
                   filter=filt)

## renew chromosomes from 'chr1.fa' to 'chr1', etc
labels <- sub("\\.fa", "", levels(chromosome(aln)))
renew(aln, chromosome=factor(chromosome(aln), labels=labels))

## multiple changes -- update chromosome, offset position
renew(aln, chromosome=factor(chromosome(aln), labels=labels),
      position=1L+position(aln))

## oops! invalid instances cannot be constructed
try(renew(aln, position=1:10))
```

report *Summarize quality assessment results into a report*

Description

This generic function summarizes results from evaluation of qa into a report. Available report formats vary depending on the data analysed.

Usage

```
report(x, ..., dest=tempfile(), type="html")
report_html(x, dest, type, ...)
```

Arguments

x	An object returned by qa, usually derived from class .QA
...	Additional arguments used by specific methods. All methods with type="html" support the argument cssFile, which is a named, length 1 character vector. The value is a path to a CSS file to be incorporated into the report (e.g., system.file("template", "QA.css", package="ShortRead")). The name of cssFile is the name of the CSS file as seen by the html report (e.g., "QA.css"). See specific methods for details on additional ... arguments.
dest	The output destination for the final report. For type="html" this is a directory; for (deprecated) type="pdf" this is a file.
type	A text string defining the type of report; available report types depend on the type of object x; usually this is "html".

Details

report_html is meant for use by package authors wishing to add methods for creating HTML reports; users should always invoke report.

The following methods are defined:

x="BowtieQA", ..., dest=tempfile(), type="html" Produce an HTML-based report from an object of class BowtieQA.

x="FastqQA", ..., dest=tempfile(), type="html" Produce an HTML-based report from an object of class FastqQA.

x="MAQMapQA", ..., dest=tempfile(), type="html" Produce an HTML-based report from an object of class MAQMapQA.

x="SolexaExportQA", ..., dest=tempfile(), type="html" Produce an HTML-based report from an object of class SolexaExportQA.

x="SolexaExportQA", ..., dest=tempfile(), type="pdf" (Deprecated) Produce an PDF report from an object of class SolexaExportQA.

x="SolexaPath", ..., dest=tempfile(), type="html" Produce an HTML report by first visiting all _export.txt files in the analysisPath directory of x to create a SolexaExportQA instance.

x="SolexaPath", ..., dest=tempfile(), type="pdf" (Deprecated) Produce an PDF report by first visiting all _export.txt files in the analysisPath directory of x to create a SolexaExportQA instance.

x="ANY", ..., dest=tempfile(), type="ANY" This method is used internally

Value

This function is invoked for its side effect; the return value is the name of the directory or file where the report was created.

Author(s)

Martin Morgan <mtmorgan@fhcrc.org>

See Also

SolexaExportQA

Examples

```
showMethods("report")

## default CSS file
cssFile <- c(QA.css=system.file("template", "QA.css",
                                package="ShortRead"))
noquote(readLines(cssFile))
```

RochePath-class *(Legacy) "RochePath" class representing a Roche (454) experiment location*

Description

This class represents the directory location where Roche (454) result files (fasta sequences and qualities) can be found.

Objects from the Class

Objects from the class are created with the RochePath constructor:

RochePath(experimentPath = NA_character_, readPath = experimentPath, qualPath = readPath, ..., verbose = FALSE)

experimentPath `character(1)` or `RochePath` pointing to the top-level directory of a Roche experiment.

readPath `character()` of directories (typically in `experimentPath`) containing sequence (read) information. The default selects all directories matching `list.files(experimentPath, "run")`.

qualPath `character()` of directories (typically in `experimentPath`) containing quality information. The default selects all directories matching `list.files(experimentPath, "run")`.

verbose `logical(1)` indicating whether invalid paths should be reported interactively.

Slots

`RocheSet` has the following slots:

`readPath`**:** Object of class `"character"`, as described in the constructor, above.

`qualPath`**:** Object of class `"character"`, as described in the constructor, above.

`basePath`**:** Object of class `"character"`, containing the `experimentPath`.

Extends

Class `"ExperimentPath"`, directly. Class `".Roche"`, directly. Class `".ShortReadBase"`, by class "ExperimentPath", distance 2. Class `".ShortReadBase"`, by class ".Roche", distance 2.

Methods

`RochePath` has the following methods or functions defined:

readFasta `signature(dirPath = "RochePath", pattern=".\.fna$", sample = 1, run = 1, ...)`: Read sequences from files matching `list.files(dirPath, pattern)` (when `dirPath="character"`) or `list.files(readPath(dir)[run], pattern)[sample]`. The result is a `DNAStringSet`.

readQual `signature(dirPath = "RochePath", reads=NULL, pattern="\.qual$", sample=1, run=1, ...)`: Read quality scores from files matching `list.files(qualPath(dirPath)[run])[sample]`. Non-null `reads` is used as an (optional) template for parsing quality scores.

readFastaQual `signature(dirPath = "RochePath", fastaPattern = "\.fna$", qualPattern = "\.qual$", sample = 1, run = 1)`: read sequences and quality scores into a `ShortReadQ` instance.

readFastaQual `signature(dirPath = "character", fastaPattern = "\.fna$", qualPattern = "\.qual$", sample = 1, run = 1)`: wrapper for method above, coercing `dirPath` to a `RochePath` via `RochePath(dirPath)`.

readBaseQuality `signature(dirPath = "RochePath", ...)`: Reads in base and quality information. Currently delegates to `readFastaQual`, above, but will do more after `RochePath` supports more file types.

read454 `signature(dirPath = "RochePath", ...)`: Pass arguments on to `readFastaQual`, documented above.

readPath `signature(object = "RochePath")`: return the contents of the `readPath` slot.

runNames `signature(object = "RochePath")`: return the basenames of `readPath(object)`.

RocheSet `signature(path = "RochePath")`: create a `RocheSet` from `path`.

Additional methods include:

show `signature(object = "RochePath")`: Briefly summarize the experiment path locations.

detail `signature(x = "RochePath")`: Provide additional detail on the Roche path. All file paths are presented in full.

Author(s)

Michael Lawrence <mflawrence@fhcrc.org>

See Also

`ExperimentPath`.

Examples

```
showClass("RochePath")
```

RocheSet-class *(Legacy) Roche (454) experiment-wide data container*

Description

This class is meant to coordinate all data in a Roche (454) experiment. See `SRSet` for additional details.

Objects from the Class

Create objects from this class using one of the `RocheSet` methods documented below

Slots

`sourcePath`**:** Object of class `"RochePath"` The file system location of the data used in this experiment.

`readIndex`**:** Object of class `"integer"` indexing reads included in the experiment; see `SRSet` for details on data representation in this class.

`readCount`**:** Object of class `"integer"` containing the number of reads associated with each sample; see `SRSet` for details on data representation in this class.

`phenoData`**:** Object of class `"AnnotatedDataFrame"` with as many rows as there are samples, containing information on experimental design.

`readData`**:** Object of class `"AnnotatedDataFrame"` containing as many rows as there are reads, containing information on each read in the experiment.

Extends

Class `"SRSet"`, directly. Class `".Roche"`, directly. Class `".ShortReadBase"`, by class "SRSet", distance 2. Class `".ShortReadBase"`, by class ".Roche", distance 2.

Methods

No methods defined with class "RocheSet" in the signature; see SRSet for inherited methods.

Author(s)

Michael Lawrence <mflawrence@fhcrc.org>

See Also

SRSet

Examples

```
showClass("RocheSet")
```

RtaIntensity *(Legacy) Construct objects of class "RtaIntensity"*

Description

RtaIntensity objects contain Illumina image intensity measures created by the RTA pipeline. It will often be more convenient to create this object using readIntensities.

Usage

```
RtaIntensity(intensity=array(0, c(0, 0, 0)),
             measurementError=array(0, c(0, 0, 0)),
             readInfo=SolexaIntensityInfo(
               lane=integer()[seq_len(nrow(intensity))]),
             ...)
```

Arguments

intensity	A matrix of image intensity values. Successive columns correspond to nucleotides A, C, G, T; four successive columns correspond to each cycle. Typically, derived from "_int.txt" files.
measurementError	As intensity, but measuring standard error. Usually derived from "_nse.txt" files.
readInfo	An object of class AnnotatedDataFrame, containing information described by RtaIntensityInfo.
...	Additional arguments, not currently used.

Value

An object of class RtaIntensity.

Author(s)

Martin Morgan <mtmorgan@fhcrc.org>

See Also

RtaIntensity, readIntensities.

Examples

```
rta <- RtaIntensity(array(runif(60), c(5,4,3)))
intensity(rta)
## subsetting, access, and coercion
as(intensity(rta)[1:2,,], "array")
```

RtaIntensity-class *(Legacy) Class "RtaIntensity"*

Description

Subclass of Intensity for representing image intensity data from the Illumina RTA pipeline.

Objects from the Class

Objects can be created by calls to RtaIntensity or more usually readIntensities.

Slots

Object of RtaIntensity have slots:

readInfo: Object of class "RtaIntensityInfo" representing information about each read.

intensity: Object of class "ArrayIntensity" containing an array of intensities with dimensions read, base, and cycle. Nucleotide are A, C, G, T for each cycle.

measurementError: Object of class "ArrayIntensity" containing measurement errors for each read, cycle, and base, with dimensions like that for intensity.

.hasMeasurementError: Object of class "ScalarLogical" used internally to indicate whether measurement error information is included.

Extends

Class "SolexaIntensity", directly.

Class "Intensity", by class "SolexaIntensity", distance 2.

Class ".ShortReadBase", by class "SolexaIntensity", distance 3.

Methods

Class "RtaIntensity" inherits accessor, subsetting, and display methods from class SolexaIntensity.

Author(s)

Martin Morgan <mtmorgan@fhcrc.org>

See Also

SolexaIntensity, readIntensities

Examples

```
showClass("RtaIntensity")
showMethods(class="RtaIntensity", where=getNamespace("ShortRead"))
```

ShortRead-class *"ShortRead" class for short reads*

Description

This class provides a way to store and manipulate, in a coordinated fashion, uniform-length short reads and their identifiers.

Objects from the Class

Objects from this class are created by readFasta, or by calls to the constructor ShortRead, as outlined below.

Slots

sread: Object of class "DNAStringSet" containing IUPAC-standard, uniform-length DNA strings represent short sequence reads.

id: Object of class "BStringSet" containing identifiers, one for each short read.

Extends

Class ".ShortReadBase", directly.

Methods

Constructors include:

ShortRead signature(sread = "DNAStringSet", id = "BStringSet"): Create a ShortRead object from reads and their identifiers. The length of id must match that of sread.

ShortRead signature(sread = "DNAStringSet", id = "missing"): Create a ShortRead object from reads, creating empty identifiers.

ShortRead signature(sread = "missing", id = "missing"): Create an empty ShortRead object.

Methods include:

sread `signature(object = "AlignedRead")`: access the sread slot of `object`. `as(object, "DNAStringSet")` is similar, but adds `id(object)` as the `names()` of the DNAStringSet.

id `signature(object = "AlignedRead")`: access the id slot of `object`.

[`signature(x = "ShortRead", i = "ANY", j = "missing")`: This method creates a new `ShortRead` object containing only those reads indexed by `i`. Additional methods on '[,ShortRead' do not provide additional functionality, but are present to limit inappropriate use.

append `signature(x = "ShortRead", values = "ShortRead")`: append the `sread` and `id` slots of `values` after the corresponding fields of `x`.

narrow `signature(x = "ShortRead", start = NA, end = NA, width = NA, use.names = TRUE)`: 'narrow' `sread` so that sequences are between `start` and `end` bases, according to `narrow` in the `IRanges` package.

length `signature(x = "ShortRead")`: returns a `integer(1)` vector describing the number of reads in this object.

width `signature(x = "ShortRead")`: returns an `integer()` vector of the widths of each read in this object.

srorder `signature(x = "ShortRead")`:

srrank `signature(x = "ShortRead")`:

srsort `signature(x = "ShortRead")`:

srduplicated `signature(x = "ShortRead")`: Order, rank, sort, and find duplicates in `ShortRead` objects based on `sread(x)`, analogous to the corresponding functions `order`, `rank`, `sort`, and `duplicated`, ordering nucleotides in the order `ACGT`.

srdistance `signature(pattern="ShortRead", subject="ANY")`: Find the edit distance between each read in `pattern` and the (short) sequences in `subject`. See `srdistance` for allowable values for `subject`, and for additional details.

trimLRPatterns `signature(Lpattern = "", Rpattern = "", subject = "ShortRead", max.Lmismatch = 0, max.Rmismatch = 0, with.Lindels = FALSE, with.Rindels = FALSE, Lfixed = TRUE, Rfixed = TRUE, ranges = FALSE)`:

Remove left and / or right flanking patterns from `sread(subject)`, as described in `trimLRPatterns`. Classes derived from `ShortRead` (e.g., `ShortReadQ`, `AlignedRead`) have corresponding base quality scores trimmed, too. The class of the return object is the same as the class of `subject`, except when `ranges=TRUE` when the return value is the ranges to use to trim 'subject'.

alphabetByCycle `signature(stringSet = "ShortRead")`: Apply `alphabetByCycle` to the `sread` component of `stringSet`, returning a matrix as described in `alphabetByCycle`.

tables `signature(x= "ShortRead", n = 50)`: Apply `tables` to the `sread` component of `x`, returning a list summarizing frequency of reads in `x`.

clean `signature(object="ShortRead")`: Remove all reads containing non-nucleotide (`"N"`, `"-"`) symbols.

show `signature(object = "ShortRead")`: provides a brief summary of the object, including its class, length and width.

detail `signature(x = "ShortRead")`: provides a more extensive summary of this object, displaying the first and last entries of `sread` and `id`.

writeFasta `signature(object, file, ...)`: write `object` to `file` in fasta format. See `writeXStringSet` for `...` argument values.

Author(s)

Martin Morgan

See Also

ShortReadQ

Examples

```
showClass("ShortRead")
showMethods(class="ShortRead", where=getNamespace("ShortRead"))
```

ShortRead-deprecated *Deprecated functions from the ShortRead package*

Description

These functions are deprecated, and will become defunct.

Usage

```
uniqueFilter(withSread=TRUE, .name="UniqueFilter")
```

Arguments

withSread	A logical(1) indicating whether uniqueness includes the read sequence (withSread=TRUE) or is based only on chromosome, position, and strand (withSread=FALSE)
.name	An optional character(1) object used to over-ride the name applied to default filters.

Details

See srFilter for details of ShortRead filters.

uniqueFilter selects elements satisfying !srduplicated(x) when withSread=TRUE, and !(duplicated(chromosome(x) & duplicated(position(x)) & duplicated(strand(x))) when withSread=FALSE.

The behavior when withSread=TRUE can be obtained with occurrenceFilter(withSread=TRUE).
The behavior when withSread=FALSE can be obtained using a custom filter

ShortReadQ-class *"ShortReadQ" class for short reads and their quality scores*

Description

This class provides a way to store and manipulate, in a coordinated fashion, the reads, identifiers, and quality scores of uniform-length short reads.

Objects from the Class

Objects from this class are the result of readFastq, or can be constructed from DNAStringSet, QualityScore, and BStringSet objects, as described below.

Slots

Slots sread and id are inherited from ShortRead. An additional slot defined in this class is:

quality: Object of class "BStringSet" representing a quality score (see readFastq for some discussion of quality score).

Extends

Class "ShortRead", directly. Class ".ShortReadBase", by class "ShortRead", distance 2.

Methods

Constructors include:

ShortReadQ signature(sread = "DNAStringSet", quality = "QualityScore", id = "BStringSet"):

ShortReadQ signature(sread = "DNAStringSet", quality = "BStringSet", id = "BStringSet"): Create a ShortReadQ object from reads, their quality scores, and identifiers. When quality is of class BStringSet, the type of encoded quality score is inferred from the letters used in the scores. The length of id and quality must match that of sread.

ShortReadQ signature(sread = "DNAStringSet", quality = "QualityScore", id = "missing"):

ShortReadQ signature(sread = "DNAStringSet", quality = "BStringSet", id = "missing"): Create a ShortReadQ object from reads and their quality scores, creating empty identifiers. When quality is of class BStringSet, the type of encoded quality score is inferred from the letters used in the scores.

ShortReadQ signature(sread = "missing", quality = "missing", id = "missing"): Create an empty ShortReadQ object.

See accessors for additional functions to access slot content, and ShortRead for inherited methods. Additional methods include:

quality inherited from signature(object = "ANY"): access the quality slot of object.

coerce `signature(from = "SFastqQuality", to = "QualityScaledDNAStringSet")`: (Use `as(from, "QualityScaledDNAStringSet")`) coerce objects of class `from` to class `to`, using the quality encoding implied by `quality(from)`. See `QualityScore` for supported quality classes and their coerced counterparts.

writeFastq `signature(object = "ShortReadQ", file = "character", ...)`:

writeFastq `signature(object = "ShortReadQ", file = "FastqFile", ...)`: Write `object` to `file` in fastq format. See ?`writeFastq` for additional arguments `mode` and `full`.

[`signature(x = "ShortReadQ", i = "ANY", j = "missing")`: This method creates a new `ShortReadQ` object containing only those reads indexed by `i`. Additional methods on '[,ShortRead' do not provide additional functionality, but are present to limit inappropriate use.

[<- `signature(x = "ShortReadQ", i = "ANY", j = "missing", ..., y="ShortReadQ")`: This method updates `x` so that records indexed by `i` are replaced by corresponding records in `value`.

append `signature(x = "ShortReadQ", values = "ShortRead")`: append the `sread`, `quality` and `id` slots of `values` after the corresponding fields of `x`.

reverse, reverseComplement `signature(x = "ShortReadQ", ...`: reverse or reverse complement the DNA sequence, and reverse the quality sequence.

narrow `signature(x = "ShortReadQ", start = NA, end = NA, width = NA, use.names = TRUE)`: narrow `sread` and `quality` so that sequences are between `start` and `end` bases, according to `narrow` in the `IRanges` package.

trimTailw `signature(object="ShortReadQ", k="integer",a="character", halfwidth="integer", ..., ranges=FALSE)`: trim trailing nucleotides when a window of width 2 * halfwidth + 1 contains `k` or more quality scores falling at or below `a`.

trimTails `signature(object="ShortReadQ", k="integer",a="character", successive=FALSE, ..., ranges=FALSE)`: trim trailing nucleotides if `k` nucleotides fall below the quality encoded by `a`. If `successive=FALSE`, the k'th failing nucleotide and all subseqent nucleotides are trimmed. If `successive=TRUE`, failing nucleotides must occur successively; the sequence is trimmed from the first of the successive failing nucleotides.

alphabetByCycle `signature(stringSet = "ShortReadQ")`: Apply `alphabetByCycle` to the `sread` component, the `quality` component, and the combination of these two components of `stringSet`, returning a list of matrices with three elements: `"sread"`, `"quality"`, and `"both"`.

alphabetScore `signature(object = "ShortReadQ")`: See `alphabetScore` for details.

qa `signature(dirPath = "ShortReadQ", lane="character",..., verbose=FALSE)`: Perform quality assessment on the `ShortReadQ` object using `lane` to identify the object and returning an instance of `ShortReadQQA`. See `qa`

detail `signature(x = "ShortReadQ")`: display the first and last entries of each of `sread`, `id`, and `quality` entries of `object`.

Author(s)

Martin Morgan

See Also

`readFastq` for creation of objects of this class from fastq-format files.

Examples

```
showClass("ShortReadQ")
showMethods(class="ShortReadQ", where=getNamespace("ShortRead"),
            inherit=FALSE)
showMethods(class="ShortRead", where=getNamespace("ShortRead"),
            inherit=FALSE)

sp <- SolexaPath(system.file('extdata', package='ShortRead'))
rfq <- readFastq(analysisPath(sp), pattern="s_1_sequence.txt")
quality(rfq)
sread(reverseComplement(rfq))
quality(reverseComplement(rfq))
quality(trimTails(rfq, 2, "H", successive=TRUE))
```

ShortReadQA-class *Quality assessment of fastq files and ShortReadQ objects*

Description

These classes contains a list-like structure with summary descriptions derived from visiting one or more fastq files, or from a ShortReadQ object.

Objects from the Class

Objects of the class are usually produced by a qa method.

Slots

`.srlist`: Object of class "list", containing data frames or lists of data frames summarizing the results of qa.

Extends

Class "SRList", directly. Class ".QA", directly. Class ".SRUtil", by class "SRList", distance 2. Class ".ShortReadBase", by class ".QA", distance 2.

Methods

Accessor methods are inherited from the SRList class.

Additional methods defined on this class are:

report `signature(x="FastqQA", ..., dest=tempfile(), type="html")`: produces HTML files summarizing QA results. `dest` should be a directory.

report `signature(x="ShortReadQA", ..., dest=tempfile(), type="html")`: produces HTML files summarizing QA results. `dest` should be a directory.

Author(s)

Martin Morgan <mtmorgan@fhcrc.org>

See Also

qa.

Examples

```
showClass("FastqQA")
```

Snapshot-class *Class* "Snapshot"

Description

A Snapshot-class to visualize genomic data from BAM files with zoom and pan functionality.

Usage

```
Snapshot(files, range, ...)
```

Arguments

files	A character() or BamFileList specifying the file(s) to be visualized.
range	A GRanges object specifying the range to be visualized.
...	Additional, optional, arguments to be passed to the Snapshot initialize function. Arguments include:

functions: A SnapshotFunctionList of functions, in addition to built-in 'fine_coverage', 'coarse_coverage', 'multifine_coverage', to be used for visualization.

currentFunction: character(1) naming the function, from functions to be used for data input and visualization. The default chooses a function based on the scale at which the data is being visualized.

annTrack: Annotation track. If built-in visualization functions are to be used, annTrack should be a GRanges instance and the first column of its elementMeatdata would be used to annotate the range.

fac: Character(1) indicating which factor used for grouping the sample files. The factor should be included in the elementMetadata of files, otherwise ignored. Used only to visualize multiple files.

.auto_display: logical(1) indicating whether the visualization is to be updated when show is invoked.

.debug logical(1) indicating whether debug messages are to be printed.

Methods

zoom signature(x = "Snapshot"): Zoom (in or out) the current plot.

pan signature(x = "Snapshot"): Pan (right or left) the current plot.

togglefun `signature(x = "Snapshot")`: Toggle the current functions which imported records are to be immediately evaluated. Note that the active range will be changed to the current active window.

togglep `signature(x = "Snapshot")`: Toggle the panning effects.

togglez `signature(x = "Snapshot")`: Toggle the zooming effects.

Accessors

show `signature(object = "Snapshot")`: Display a `Snapshot` object.

files `signature(x = "Snapshot")`: Get the `files` field (object of class `BamFileList`) of a `Snapshot` object.

functions `signature(x = "Snapshot")`: Get the `functions` field (object of `SnapshotFunctionList`) of a `Snapshot` object.

view `signature(x = "Snapshot")`: Get the `view` field (object of `SpTrellis`) of a `Snapshot` object.

vrange `signature(x = "Snapshot")`: Get the `.range` field (object of `GRanges`) of a `Snapshot` object.

getTrellis `signature(x = "Snapshot")`: Get the `trellis` object, a field of the `SpTrellis` object.

Fields

`.debug`**:** Object of class `function` to display messages while in debug mode

`.auto_display`**:** Object of class `logical` to automatically display the coverage plot.

`.range`**:** Object of class `GRanges` indicating which ranges of records to be imported from BAM fields.

`.zin`**:** Object of class `logical` indicating whether the current zooming effect is zoom in.

`.pright`**:** Object of class `logical` indicating whether the current panning effect is right.

`.data`**:** Object of class `data.frame` containing coverage a position is represented for each strand and BAM file.

`.data_dirty`**:** Object of class `logical` indicating whether to re-evaluate the imported records.

`.initial_functions`**:** Object of class `SnapshotFunctionList` available by the `Snapshot` object.

`.current_function`**:** Object of class `character` of the function the imported recorded are currently evaluated and visualized.

`annTrack`**:** Default to `NULL` if not intended to visualize the annotation track. If default visualization function(s) is intended to be used to plot the annotation, `annTrack` has to be a `GRanges` instance.

`functions`**:** Object of class `SnapshotFunctionList` of customized functions to evaluate and visualize the imported records.

`files`**:** Object of class `BamFileList` to be imported.

`view`**:** Object of class `SpTrellis` that is essentially a reference class wrapper of `Trellis` objects.

Class-Based Methods

`display()`: Display the current `Snapshot` object.

`pan()`: Pan (right or left) the current plot.

`zoom()`: Zoom (in or out) the current plot.

`toggle(zoom, pan, currentFunction)`: Toggle zooming, panning effects or the currentFuction in which the imported records are to be evaluated and visualized.

Author(s)

Martin Morgan and Chao-Jen Wong <cwon2@fhcrc.org>

See Also

SpTrellis

Examples

```
## example 1: Importing specific ranges of records

file <- system.file("extdata", "SRR002051.chrI-V.bam",
                    package="yeastNagalakshmi")
which <-  GRanges("chrI", IRanges(1, 2e5))
s <- Snapshot(file, range=which)

## methods
zoom(s) # zoom in
## zoom in to a specific region
zoom(s, range=GRanges("chrI", IRanges(7e4, 7e4+8000)))
pan(s)  # pan right
togglez(s) # change effect of zooming
zoom(s) # zoom out
togglep(s) # change effect of panning
pan(s)

## accessors
functions(s)
vrange(s)
show(s)
ignore.strand(s)
view(s) ## extract the spTrellis object
getTrellis(s) ## extract the trellis object

## example 2: ignore strand
s <- Snapshot(file, range=which, ignore.strand=TRUE)

##
## example 3: visualizing annotation track
##

library(GenomicFeatures)
```

```
getAnnGR <- function(txdb, which) {
    ex <- exonsBy(txdb, by="gene")
    seqlevels(ex, pruning.mode="coarse") <- seqlevels(which)
    r <- range(ex)
    gr <- unlist(r)
    values(gr)[["gene_id"]] <- rep.int(names(r), times=lengths(r))
    gr
}

txdbFile <- system.file("extdata", "sacCer2_sgdGene.sqlite",
                    package="yeastNagalakshmi")
# txdb <- makeTxDbFromUCSC(genome="sacCer2", tablename="sgdGene")
txdb <- loadDb(txdbFile)
which <-  GRanges("chrI", IRanges(1, 2e5))
gr <- getAnnGR(txdb, which)
## note that the first column of the elementMetadata annotates of the
## range of the elements.
gr

s <- Snapshot(file, range=which, annTrack=gr)
annTrack(s)
## zoom in to an interesting region
zoom(s, range=GRanges("chrI", IRanges(7e4, 7e4+8000)))

togglez(s) ## zoom out
zoom(s)

pan(s)

## example 4, 5, 6: multiple BAM files with 'multicoarse_covarage'
## and 'multifine_coverage' view.

## Resolution does not automatically switch for views of multiple
## files. It is important to note if width(which) < 10,000, use
## multifine_coverage.  Otherwise use multicoarse_coverage
file <- system.file("extdata", "SRR002051.chrI-V.bam",
                    package="yeastNagalakshmi")
which <-  GRanges("chrI", IRanges(1, 2e5))
s <- Snapshot(c(file, file), range=which,
              currentFunction="multicoarse_coverage")

## grouping files and view by 'multicoarse_coverage'
bfiles <- BamFileList(c(a=file, b=file))
values(bfiles) <- DataFrame(sampleGroup=factor(c("normal", "tumor")))
values(bfiles)
s <- Snapshot(bfiles, range=which,
              currentFunction="multicoarse_coverage", fac="sampleGroup")

## grouping files and view by 'multifine_coverage'
which <- GRanges("chrI", IRanges(7e4, 7e4+8000))
s <- Snapshot(bfiles, range=which,
              currentFunction="multifine_coverage", fac="sampleGroup")
```

SnapshotFunction-class
Class "SnapshotFunction"

Description

A class to store custom reader and viewer functions for the Snapshot class.

Usage

```
SnapshotFunction(reader, viewer, limits, ...)
reader(x, ...)
viewer(x, ...)
limits(x, ...)
```

Arguments

reader	A function for reading data. The function must take a single argument (a Snapshot instance) and return a data.frame summarizing the file.
viewer	A function for visualizing the data. The function must accept the data.frame created by reader, and return an SpTrellis object representing the view.
limits	An integer(2) indicating the minimum and maximum number of nucleotides the SnapshotFunction is intended to visualize. For instance, a 'fine-scale' viewer displaying a pileup might be appropriate at between 1000 and 50000 nucleotides.
x	An instance of SnapshotFunction
...	Additional arguments, currently unused.

Fields

reader: Object of class function for reading data from BAM files and returning a data.frame.

viewer: Object of class function for visualization that returns an SpTrellis object.

limits: Object of class integer for the limits of ranges to be visualized.

Author(s)

Martin Morgan and Chao-Jen Wong

See Also

Snapshot

Examples

```
## internally defined function
reader(ShortRead:::.fine_coverage)
viewer(ShortRead:::.fine_coverage)
limits(ShortRead:::.fine_coverage)
```

SolexaExportQA-class *(Legacy) Quality assessment summaries from Solexa export and realign files*

Description

This class contains a list-like structure with summary descriptions derived from visiting one or more Solexa 'export' or 'realign' files.

Objects from the Class

Objects of the class are usually produced by a qa method.

Slots

`.srlist`: Object of class "list", containing data frames or lists of data frames summarizing the results of qa.

Extends

Class "SRList", directly. Class ".QA", directly. Class ".SRUtil", by class "SRList", distance 2. Class ".ShortReadBase", by class ".QA", distance 2.

Methods

Accessor methods are inherited from the SRList class.

Additional methods defined on this class are:

report `signature(x="SolexaExportQA", ..., dest=tempfile(), type="html")`: produces HTML files summarizing QA results. `dest` should be a directory.

report `signature(x="SolexaExportQA", ..., dest=tempfile(), type="pdf")`: (deprecated; use `type="html"` instead) produces a pdf file summarizing QA results. `dest` should be a file.

report `signature(x="SolexaRealignQA", ..., dest=tempfile(), type="html")`: produces HTML files summarizing QA results. `dest` should be a directory.

Author(s)

Martin Morgan <mtmorgan@fhcrc.org>

See Also

qa.

Examples

```
showClass("SolexaExportQA")
```

`SolexaIntensity` *(Legacy) Construct objects of class "SolexaIntensity" and "SolexaIntensityInfo"*

Description

These function constructs objects of `SolexaIntensity` and `SolexaIntensityInfo`. It will often be more convenient to create these objects using parsers such as `readIntensities`.

Usage

```
SolexaIntensity(intensity=array(0, c(0, 0, 0)),
                measurementError=array(0, c(0, 0, 0)),
                readInfo=SolexaIntensityInfo(
                  lane=integer(nrow(intensity))),
                ...)
SolexaIntensityInfo(lane=integer(0),
                    tile=integer(0)[seq_along(lane)],
                    x=integer(0)[seq_along(lane)],
                    y=integer(0)[seq_along(lane)])
```

Arguments

`intensity`	A matrix of image intensity values. Successive columns correspond to nucleotides A, C, G, T; four successive columns correspond to each cycle. Typically, derived from `"_int.txt"` files.
`measurementError`	As `intensity`, but measuring standard error. Usually derived from `"_nse.txt"` files.
`readInfo`	An object of class `AnnotatedDataFrame`, containing information described by `SolexaIntensityInfo`.
`lane`	An integer vector giving the lane from which each read is derived.
`tile`	An integer vector giving the tile from which each read is derived.
`x`	An integer vector giving the tile-local x coordinate of the read from which each read is derived.
`y`	An integer vector giving the tile-local y coordinate of the read from which each read is derived.
`...`	Additional arguments, not currently used.

Value

An object of class `SolexaIntensity`, or `SolexaIntensityInfo`.

Author(s)

Martin Morgan <mtmorgan@fhcrc.org>

See Also

SolexaIntensity.

SolexaIntensity-class *Classes "SolexaIntensity" and "SolexaIntensityInfo"*

Description

Instances of Intensity and IntensityInfo for representing image intensity data from Solexa experiments.

Objects from the Class

Objects can be created by calls to SolexaIntensityInfo or SolexaIntensity, or more usually readIntensities.

Slots

Object of SolexaIntensity have slots:

readInfo: Object of class "SolexaIntensityInfo" representing information about each read.

intensity: Object of class "ArrayIntensity" containing an array of intensities with dimensions read, base, and cycle. Nucleotide are A, C, G, T for each cycle.

measurementError: Object of class "ArrayIntensity" containing measurement errors for each read, cycle, and base, with dimensions like that for intensity.

.hasMeasurementError: Object of class "ScalarLogical" used internally to indicate whether measurement error information is included.

Object of SolexaIntensityInfo

data Object of class "data.frame", inherited from AnnotatedDataFrame.

varMetadata Object of class "data.frame", inherited from AnnotatedDataFrame.

dimLabels Object of class "character", inherited from AnnotatedDataFrame.

.__classVersion__ Object of class "Versions", inherited from AnnotatedDataFrame.

.init Object of class "ScalarLogical", used internally to indicate whether the user initialized this object.

Extends

Class SolexaIntensity:

Class "Intensity", directly. Class ".ShortReadBase", by class "Intensity", distance 2.

Class SolexaIntensityInfo:

Class "AnnotatedDataFrame", directly Class "IntensityInfo", directly Class "Versioned", by class "AnnotatedDataFrame", distance 2 Class ".ShortReadBase", by class "IntensityInfo", distance 2 Class "IntensityInfo", directly.

Methods

Class "SolexaIntensity" inherits accessor and display methods from class Intensity. Additional methods include:

[signature(x = "SolexaIntensity", i="ANY", j="ANY", k="ANY"):
Selects the ith read, jth nucleotide, and kth cycle. Selection is coordinated across intensity, measurement error, and read information.

Class "SolexaIntensityInfo" inherits accessor, subsetting, and display methods from class IntensityInfo and AnnotatedDataFrame.

Author(s)

Martin Morgan <mtmorgan@fhcrc.org>

See Also

readIntensities

Examples

```
showClass("SolexaIntensity")
sp <- SolexaPath(system.file('extdata', package='ShortRead'))
int <- readIntensities(sp)
int     # SolexaIntensity
readIntensityInfo(int)  # SolexaIntensityInfo
int[1:5,,]   # read 1:5
```

SolexaPath-class *(Legacy) "SolexaPath" class representing a standard output file hierarchy*

Description

Solexa produces a hierarchy of output files. The content of the hierarchy varies depending on analysis options. This class represents a standard class hierarchy, constructed by searching a file hierarchy for appropriately named directories.

Objects from the Class

Objects from the class are created by calls to the constructor:

```
SolexaPath(experimentPath, dataPath=.solexaPath(experimentPath, "Data"), scanPath=.solexaPath(dataPat
"GoldCrest"), imageAnalysisPath=.solexaPath(dataPath, "^(C|IPAR)"), baseCallPath=.solexaPath(imageAna
"^Bustard"), analysisPath=.solexaPath(baseCallPath, "^GERALD"), ..., verbose=FALSE)
```

experimentPath `character(1)` object pointing to the top-level directory of a Solexa run, e.g., `/home/solexa/user/080220_HWI-EAS88_0004`. This is the only required argument

dataPath (optional) Solexa 'Data' folder .

scanPath (optional) Solexa GoldCrest image scan path.

imageAnalysisPath (optional) Firecrest image analysis path.

baseCallPath (optional) Bustard base call path.

analysisPath (optional) Gerald analysis pipeline path.

... Additional arguments, unused by currently implemented methods.

verbose=FALSE (optional) logical vector which, when `TRUE` results in warnings if paths do not exist.

All paths must be fully-specified.

Slots

`SolexaPath` has the following slots, containing either a fully specified path to the corresponding directory (described above) or `NA` if no appropriate directory was discovered.

`basePath` See `experimentPath`, above.

`dataPath` See above.

`scanPath` See above.

`imageAnalysisPath` See above.

`baseCallPath` See above.

`analysisPath` See above.

Extends

Class "`.Solexa`", directly. Class "`.ShortReadBase`", by class ".Solexa", distance 2.

Methods

Transforming methods include:

readIntensities `signature(dirPath = "SolexaPath", pattern=character(0), run, ...)`: Use `imageAnalysisPath(sp)[run]` as the directory path(s) and `pattern=character(0)` as the pattern for discovering Solexa intensity files. See `readIntensities,character-method` for additional parameters.

readPrb `signature(dirPath = "SolexaPath", pattern=character(0), run, ...)`:
Use `baseCallPath(dirPath)[run]` as the directory path(s) and `pattern=character(0)` as the pattern for discovering Solexa 'prb' files, returning a `SFastqQuality` object containing the maximum qualities found for each base of each cycle.

The `...` argument may include the named argument `as`. This influences the return value, as explained on the `readPrb,character-method` page.

readFasta `signature(dirPath, pattern = character(0), ..., nrec=-1L, skip=0L)`:
Use `analysisPath(dirPath)[run]` as the directory path(s) for discovering fasta-formatted files, returning a `ShortRead` object. The default method reads *all* files into a single object.

readFastq `signature(dirPath = "SolexaPath", pattern = ".*_sequence.txt",run, ..., qualityType="SFastqQu`
Use `analysisPath(dirPath)[run]` as the directory path(s) and `pattern=".*_sequence.txt"` as the pattern for discovering fastq-formatted files, returning a `ShortReadQ` object. The default method reads *all* sequence files into a single object.

readBaseQuality `signature(dirPath = "SolexaPath", seqPattern = ".*_seq.txt", prbPattern = "s_[1-8]_prb.txt", run, ...)`:
Use `baseCallPath(dirPath)[run]` as the directory path(s) and `seqPattern=".*_seq.txt"` as the pattern for discovering base calls and `prbPattern=".*_prb.txt"` as the pattern for discovering quality scores. Note that the default method reads *all* base call and quality score files into a single object; often one will want to specify a pattern for each lane.

readQseq `signature(directory="SolexaPath", pattern=".*_qseq.txt.*", run, ...., filtered=FALSE)`:
Use `analysisPath(dirPath)[run]` as the directory path and `pattern=".*_qseq.txt.*"` as the pattern for discovering read and quality scores in Solexa 'qseq' files. Data from *all* files are read into a single object; often one will want to specify a pattern for each lane. Details are as for `readQseq,character-method`.

readAligned `signature(dirPath = "SolexaPath", pattern = ".*_export.txt.*", run, ..., filter=srFilter())`:
Use `analysisPath(dirPath)[run]` as the directory path and `pattern=".*_export.txt"` as the pattern for discovering Eland-aligned reads in the Solexa 'export' file format. Note that the default method reads *all* aligned read files into a single object; often one will want to specify a pattern for each lane. Use an object of `SRFilter` to select specific chromosomes, strands, etc.

qa `signature(dirPath="SolexaPath", pattern="character(0)", run, ...)`:
Use `analysisPath(dirPath)[run]` as the directory path(s) and `pattern=".*_export.txt"` as the pattern for discovering Solexa `export`-formatted fileds, returning a `SolexaExportQA` object summarizing quality assessment. If `Rmpi` or `parallel` has been initiated, quality assessment calculations are distributed across available nodes or cores (one node per export file.)

report `signature(x, ..., dest=tempfile(), type="pdf")`: Use `qa(x, ...)` to generate quality assessment measures, and use these to generate a quality assessment report at location `dest` of type `type` (e.g., 'pdf').

SolexaSet `signature(path = "SolexaPath")`: create a `SolexaSet` object based on `path`.

Additional methods include:

show `signature(object = "SolexaPath")`: briefly summarize the file paths of `object`. The `experimentPath` is given in full; the remaining paths are identified by their leading characters.

detail `signature(x = "SolexaPath")`: summarize file paths of `x`. All file paths are presented in full.

Author(s)

Martin Morgan

Examples

```
showClass("SolexaPath")
showMethods(class="SolexaPath", where=getNamespace("ShortRead"))
sf <- system.file("extdata", package="ShortRead")
sp <- SolexaPath(sf)
sp
readFastq(sp, pattern="s_1_sequence.txt")
## Not run:
nfiles <- length(list.files(analysisPath(sp), "s_[1-8]_export.txt"))
library(Rmpi)
mpi.spawn.Rslaves(nslaves=nfiles)
report(qa(sp))

## End(Not run)
## Not run:
nfiles <- length(list.files(analysisPath(sp), "s_[1-8]_export.txt"))
report(qa(sp))

## End(Not run)
```

SolexaSet-class *(Legacy) "SolexaSet" coordinating Solexa output locations with sample annotations*

Description

This class coordinates the file hierarchy produced by the Solexa 'pipeline' with annotation data contained in an `AnnotatedDataFrame` (defined in the **Biobase** package).

Objects from the Class

Objects can be created from the constructor:

`SolexaSet(path, ...)`.

path A `character(1)` vector giving the fully-qualified path to the root of the directory hierarchy associated with each Solexa flow cell, or an object of class `SolexaPath` (see `SolexaPath` for this method).

... Additional arguments, especially `laneDescription`, an `AnnotatedDataFrame` describing the content of each of the 8 lanes in the Solexa flow cell.

Slots

SolexaSet has the following slots:

solexaPath: Object of class "SolexaPath".

laneDescription: Object of class "AnnotatedDataFrame", containing information about the samples in each lane of the flow cell.

Extends

Class ".Solexa", directly. Class ".ShortReadBase", by class ".Solexa", distance 2.

Methods

solexaPath signature(object = "SolexaSet"): Return the directory paths present when this object was created as a SolexaPath.

laneNames signature(object = "SolexaSet"): Return the names of each lane in the flow cell, currently names are simply 1:8.

show signature(object = "SolexaSet"): Briefly summarize the experiment path and lane description of the Solexa set.

detail signature(x = "SolexaSet"): Provide additional detail on the Solexa set, including the content of solexaPath and the pData and varMetadata of laneDescription.

Methods transforming SolexaSet objects include:

readAligned signature(dirPath = "SolexaSet", pattern = ".*_export.txt", run, ..., filter=srFilter()): Use analysisPath(solexaPath(dirPath))[run] as the directory path(s) and pattern=".*_export.txt" as the pattern for discovering Eland-aligned reads in the Solexa 'export' file format. Note that the default method reads *all* aligned read files into a single object; often one will want to specify a pattern for each lane. Use an object of SRFilter to select specific chromosomes, strands, etc.

Author(s)

Martin Morgan

Examples

```
showClass("SolexaSet")
showMethods(class="SolexaSet", where=getNamespace("ShortRead"))
## construct a SolexaSet
sf <- system.file("extdata", package="ShortRead")
df <- data.frame(Sample=c("Sample 1", "Sample 2", "Sample 3", "Sample
                          4", "Center-wide control", "Sample 6", "Sample
                          7", "Sample 8"),
                 Genome=c(rep("hg18", 4), "phi_plus_SNPs.txt",
                          rep("hg18", 3)))
dfMeta <- data.frame(labelDescription=c("Type of sample",
                                        "Alignment genome"))
adf <- new("AnnotatedDataFrame", data=df, varMetadata=dfMeta)
SolexaSet(sf, adf)
```

SpTrellis-class *Class "SpTrellis"*

Description

A reference class to manage the trellis graphics related component of the Snapshot functionality for visualization of genomic data.

Usage

```
SpTrellis(trellis, debug_enabled=FALSE)
```

Arguments

trellis	A trellis object for storing the plot of the genome area being visualized.
debug_enabled	logical(1) indicating whether class methods should report debugging information to the user.

Fields

trellis: Object of class trellis for storing the plot information.

debug_enabled logical(1) indicating whether class methods should report debugging information to the user.

Methods

zi signature(x="SpTrellis"): zoom in

zo signature(x="SpTrellis"): zoom out

right signature(x="SpTrellis"): shift to the right

left signature(x="SpTrellis"): shift to the left

restore signature(x="SpTrellis"): restore to the original plot

show signature(x="SpTrellis"): show the current plot

update signature(x="SpTrellis"): update the trellis parameters of the SpTrellis object.

Author(s)

Chao-Jen cwon2@fhcrc.org

See Also

Snapshot

Examples

```
col <- c("#66C2A5", "#FC8D62")
x = numeric(1000)
x[sample(1000, 100)] <- abs(rnorm(100))
df <- data.frame(x = c(x, -x), pos = seq(1, 1e5, length.out=1000),
                 group = rep(c("positive", "negative"), each=1000))
cv <- lattice::xyplot(x ~ pos, df, group=group, type="s",
    col=col, main="yeast chrI:1 - 2e5",
    ylab="Coverage", xlab="Coordinate",
    scales=list(y=list(tck=c(1,0)),
                x=list(rot=45, tck=c(1,0), tick.number=20)),
    panel=function(...) {
            lattice::panel.xyplot(...)
            lattice::panel.grid(h=-1, v=20)
            lattice::panel.abline(a=0, b=0, col="grey")
    })
s <- SpTrellis(cv)
s
zi(s)
zi(s)
left(s)
right(s)
zo(s)
restore(s)
```

`spViewPerFeature` *Tools to visualize genomic data*

Description

Use `Snapshot`-class to visualize a specific region of genomic data

Usage

```
spViewPerFeature(GRL, name, files, ignore.strand=FALSE,
                 multi.levels = FALSE, fac=character(0L), ...)
```

Arguments

`GRL`	Object `GRangeList` containing annotation of genomic data. It can be generated by applying `exonsBy()` or `transcriptsBy()` to a `TxDb` instance. See examples below.
`name`	Character(1) specifying which element in `GRL` to be visualized.
`files`	Charactor() or `BamFileList` specifying the file(s) to be visualized. If multiple files, local metadata of the files can be hold by setting a `DataFrame` (values(files) <- DataFrame(...)). See examples below.
`ignore.strand`	Logical(1) indicating whether to ignore the strand of the genomic data.

multi.levels	Logical(1) indicating whether to plot the coverage of multiple files on different panels. If FALSE, the mean coverage of multiple files would be plotted.
fac	Character(1) indicating which column of local metadata (elementMetatdata()) should be used to group the samples. Ignore
...	Arguments used for creating a Snapshot object.

Value

A Snapshot instance

Author(s)

Chao-Jen Wong <cwon2@fhcrc.org>

See Also

Snapshot

Examples

```
## Example 1
library(GenomicFeatures)
txdbFile <- system.file("extdata", "sacCer2_sgdGene.sqlite",
                        package="yeastNagalakshmi")

## either use a txdb file quaried from UCSC or use existing TxDb packages.
txdb <- loadDb(txdbFile)

grl <- exonsBy(txdb, by="gene")
file <- system.file("extdata", "SRR002051.chrI-V.bam",
                        package="yeastNagalakshmi")
s <- spViewPerFeature(GRL=grl, name="YAL001C", files=file)

## Example 2
## multi-files: using 'BamFileList' and setting up the 'DataFrame'
## holding the phenotype data

bfiles <- BamFileList(c(a=file, b=file))
values(bfiles) <- DataFrame(sampleGroup=factor(c("normal", "tumor")))
values(bfiles)

s <- spViewPerFeature(GRL=grl, name="YAL001C",
                      files=bfiles, multi.levels=TRUE, fac="sampleGroup")
```

srdistance *Edit distances between reads and a small number of short references*

Description

srdistance calculates the edit distance from each read in pattern to each read in subject. The underlying algorithm pairwiseAlignment is only efficient when both reads are short, and when the number of subject reads is small.

Usage

```
srdistance(pattern, subject, ...)
```

Arguments

pattern	An object of class DNAStringSet containing reads whose edit distance is desired.
subject	A short character vector, DNAString or (small) DNAStringSet to serve as reference.
...	additional arguments, unused.

Details

The underlying algorithm performs pairwise alignment from each read in pattern to each sequence in subject. The return value is a list of numeric vectors of distances, one list element for each sequence in subject. The vector in each list element contains for each read in pattern the edit distance from the read to the corresponding subject. The weight matrix and gap penalties used to calculate the distance are structured to weight base substitutions and single base insert/deletions equally. Edit distance between known and ambiguous (e.g., N) nucleotides, or between ambiguous nucleotides, are weighted as though each possible nucleotide in the ambiguity were equally likely.

Value

A list of length equal to that of subject. Each element is a numeric vector equal to the length of pattern, with values corresponding to the minimum distance between between the corresponding pattern and subject sequences.

Author(s)

Martin Morgan <mtmorgan@fhcrc.org>

See Also

pairwiseAlignment

Examples

```
sp <- SolexaPath(system.file("extdata", package="ShortRead"))
aln <- readAligned(sp, "s_2_export.txt")
polyA <- polyn("A", 35)
polyT <- polyn("T", 35)

d1 <- srdistance(clean(sread(aln)), polyA)
d2 <- srdistance(sread(aln), polyA)
d3 <- srdistance(sread(aln), c(polyA, polyT))
```

---

srduplicated *Order, sort, and find duplicates in XStringSet objects*

Description

These generics order, rank, sort, and find duplicates in short read objects, including fastq-encoded qualities. `srorder`, `srrank` and `srsort` differ from the default functions `rank`, `order` and `sort` in that sorting is based on an internally-defined order rather than, e.g., the order implied by `LC_COLLATE`.

Usage

```
srorder(x, ...)
srrank(x, ...)
srsort(x, ...)
srduplicated(x, ...)
```

Arguments

x	The object to be sorted, ranked, ordered, or to have duplicates identified; see the examples below for objects for which methods are defined.
...	Additional arguments available for use by methods; usually ignored.

Details

Unlike `sort` and friends, the implementation does not preserve order of duplicated elements. Like `duplicated`, one element in each set of duplicates is marked as `FALSE`.

`srrank` settles ties using the "min" criterion described in `rank`, i.e., identical elements are ranked equal to the rank of the first occurrence of the sorted element.

The following methods are defined, in addition to methods described in class-specific documentation:

srsort `signature(x = "XStringSet")`:

srorder `signature(x = "XStringSet")`:

srduplicated `signature(x = "XStringSet")`:

Apply `srorder`, `srrank`, `srsort`, `srduplicated` to `XStringSet` objects such as those returned by `sread`.

srsort `signature(x = "ShortRead")`:

srorder `signature(x = "ShortRead")`:

srduplicated `signature(x = "ShortRead")`:

Apply `srorder`, `srrank`, `srsort`, `srduplicated` to `XStringSet` objects to the `sread` component of `ShortRead` and derived objects.

Value

The functions return the following values:

`srorder`	An integer vector the same length as `x`, containing the indices that will bring `x` into sorted order.
`srrank`	An integer vector the same length as `x`, containing the rank of each seqeunce when sorted.
`srsort`	An instance of `x` in sorted order.
`srduplicated`	A logical vector the same length as `x` indicating whether the indexed element is already present. Note that, like `duplicated`, subsetting `x` using the result returned by `!srduplicated(x)` includes one representative from each set of duplicates.

Author(s)

Martin Morgan <mtmorgan@fhcrc.org>

Examples

```
showMethods("srsort")
showMethods("srorder")
showMethods("srduplicated")

sp <- SolexaPath(system.file('extdata', package='ShortRead'))
rfq <- readFastq(analysisPath(sp), pattern="s_1_sequence.txt")

sum(srduplicated(sread(rfq)))
srsort(sread(rfq))
srsort(quality(rfq))
```

---

srFilter *Functions for user-created and built-in ShortRead filters*

Description

These functions create user-defined (`srFitler`) or built-in instances of `SRFilter` objects. Filters can be applied to objects from `ShortRead`, returning a logical vector to be used to subset the objects to include only those components satisfying the filter.

Usage

```
srFilter(fun, name = NA_character_, ...)
## S4 method for signature 'missing'
srFilter(fun, name=NA_character_, ...)
## S4 method for signature 'function'
srFilter(fun, name=NA_character_, ...)

compose(filt, ..., .name)

idFilter(regex=character(0), fixed=FALSE, exclude=FALSE,
         .name="idFilter")
occurrenceFilter(min=1L, max=1L,
                 withSread=c(NA, TRUE, FALSE),
                 duplicates=c("head", "tail", "sample", "none"),
                 .name=.occurrenceName(min, max, withSread,
                                       duplicates))
nFilter(threshold=0L, .name="CleanNFilter")
polynFilter(threshold=0L, nuc=c("A", "C", "T", "G", "other"),
           .name="PolyNFilter")
dustyFilter(threshold=Inf, batchSize=NA, .name="DustyFilter")
srdistanceFilter(subject=character(0), threshold=0L,
                 .name="SRDistanceFilter")

##
## legacy filters for ungapped alignments
##

chromosomeFilter(regex=character(0), fixed=FALSE, exclude=FALSE,
                 .name="ChromosomeFilter")
positionFilter(min=-Inf, max=Inf, .name="PositionFilter")
strandFilter(strandLevels=character(0), .name="StrandFilter")
alignQualityFilter(threshold=0L, .name="AlignQualityFilter")
alignDataFilter(expr=expression(), .name="AlignDataFilter")
```

Arguments

fun	An object of class function to be used as a filter. fun must accept a single named argument x, and is expected to return a logical vector such that x[fun(x)] selects only those elements of x satisfying the conditions of fun
name	A character(1) object to be used as the name of the filter. The name is useful for debugging and reference.
filt	A SRFilter object, to be used with additional arguments to create a composite filter.
.name	An optional character(1) object used to over-ride the name applied to default filters.
regex	Either character(0) or a character(1) regular expression used as grep(regex, chromosome(x)) to filter based on chromosome. The default (character(0))

	performs no filtering
fixed	logical(1) passed to grep, influencing how pattern matching occurs.
exclude	logical(1) which, when TRUE, uses regex to exclude, rather than include, reads.
min	numeric(1)
max	numeric(1). For positionFilter, min and max define the closed interval in which position must be found min <= position <= max. For occurrenceFilter, min and max define the minimum and maximum number of times a read occurs after the filter.
strandLevels	Either character(0) or character(1) containing strand levels to be selected. ShortRead objects have standard strand levels NA, "+", "-", "*", with NA meaning strand information not available and "*" meaning strand information not relevant.
withSread	A logical(1) indicating whether uniqueness includes the read sequence (withSread=TRUE), is based only on chromosome, position, and strand (withSread=FALSE), or only the read sequence (withSread=NA), as described for occurrenceFilter below..
duplicates	Either character{1}, a function name, or a function taking a single argument. Influence how duplicates are handled, as described for occurrenceFilter below.
threshold	A numeric(1) value representing a minimum (srdistanceFilter, alignQualityFilter) or maximum (nFilter, polynFilter, dustyFilter) criterion for the filter. The minima and maxima are closed-interval (i.e., x >= threshold, x <= threshold for some property x of the object being filtered).
nuc	A character vector containing IUPAC symbols for nucleotides or the value "other" corresponding to all non-nucleotide symbols, e.g., N.
batchSize	NA or an integer(1) vector indicating the number of DNA sequences to be processed simultaneously by dustyFilter. By default, all reads are processed simultaneously. Smaller values use less memory but are computationally less efficient.
subject	A character() of any length, to be used as the corresponding argument to srdistance.
expr	A expression to be evaluated with pData(alignData(x)).
...	Additional arguments for subsequent methods; these arguments are not currently used.

Details

srFilter allows users to construct their own filters. The fun argument to srFilter must be a function accepting a single argument x and returning a logical vector that can be used to select elements of x satisfying the filter with x[fun(x)]

The signature(fun="missing") method creates a default filter that returns a vector of TRUE values with length equal to length(x).

compose constructs a new filter from one or more existing filter. The result is a filter that returns a logical vector with indices corresponding to components of x that pass all filters. If not provided, the name of the filter consists of the names of all component filters, each separated by " o ".

The remaining functions documented on this page are built-in filters that accept an argument x and return a logical vector of length(x) indicating which components of x satisfy the filter.

idFilter selects elements satisfying grep(regex, id(x), fixed=fixed).

chromosomeFilter selects elements satisfying grep(regex, chromosome(x), fixed=fixed).

positionFilter selects elements satisfying min <= position(x) <= max.

strandFilter selects elements satisfying match(strand(x), strand, nomatch=0) > 0.

occurrenceFilter selects elements that occur >=min and <=max times. withSread determines how reads will be treated: TRUE to include the sread, chromosome, strand, and position when determining occurrence, FALSE to include chromosome, strand, and position, and NA to include only sread. The default is withSread=NA. duplicates determines how reads with more than max reads are treated. head selects the first max reads of each set of duplicates, tail the last max reads, and sample a random sample of max reads. none removes all reads represented more than max times. The user can also provide a function (as used by tapply) of a single argument to select amongst reads.

nFilter selects elements with fewer than threshold 'N' symbols in each element of sread(x).

polynFilter selects elements with fewer than threshold copies of any nucleotide indicated by nuc.

dustyFilter selects elements with high sequence complexity, as characterized by their dustyScore. This emulates the dust command from WindowMaker software. Calculations can be memory intensive; use batchSize to process the argument to dustyFilter in batches of the specified size.

srdistanceFilter selects elements at an edit distance greater than threshold from all sequences in subject.

alignQualityFilter selects elements with alignQuality(x) greater than threshold.

alignDataFilter selects elements with pData(alignData(x)) satisfying expr. expr should be formulated as though it were to be evaluated as eval(expr, pData(alignData(x))).

Value

srFilter returns an object of SRFilter.

Built-in filters return a logical vector of length(x), with TRUE indicating components that pass the filter.

Author(s)

Martin Morgan <mtmorgan@fhcrc.org>

See Also

SRFilter.

Examples

```
sp <- SolexaPath(system.file("extdata", package="ShortRead"))
aln <- readAligned(sp, "s_2_export.txt") # Solexa export file, as example

# a 'chromosome 5' filter
```

```
filt <- chromosomeFilter("chr5.fa")
aln[filt(aln)]
# filter during input
readAligned(sp, "s_2_export.txt", filter=filt)

# x- and y- coordinates stored in alignData, when source is SolexaExport
xy <- alignDataFilter(expression(abs(x-500) > 200 & abs(y-500) > 200))
aln[xy(aln)]

# both filters as a single filter
chr5xy <- compose(filt, xy)
aln[chr5xy(aln)]

# both filters as a collection
filters <- c(filt, xy)
subsetByFilter(aln, filters)
summary(filters, aln)

# read, chromosome, strand, position tuples occurring exactly once
aln[occurrenceFilter(withSread=TRUE, duplicates="none")(aln)]
# reads occurring exactly once
aln[occurrenceFilter(withSread=NA, duplicates="none")(aln)]
# chromosome, strand, position tuples occurring exactly once
aln[occurrenceFilter(withSread=FALSE, duplicates="none")(aln)]

# custom filter: minimum calibrated base call quality >20
goodq <- srFilter(function(x) {
    apply(as(quality(x), "matrix"), 1, min, na.rm=TRUE) > 20
}, name="GoodQualityBases")
goodq
aln[goodq(aln)]
```

SRFilter-class *"SRFilter" for representing functions operating on ShortRead objects*

Description

Objects of this class are functions that, when provided an appropriate object from the ShortRead package, return logical vectors indicating which parts of the object satisfy the filter criterion.

A number of filters are built-in (described below); users are free to create their own filters, using the `srFilter` function.

Objects from the Class

Objects can be created through `srFilter` (to create a user-defined filter) or through calls to constructors for predefined filters, as described on the `srFilter` page.

Slots

`.Data`: Object of class `"function"` taking a single named argument `x` corresponding to the ShortRead object that the filter will be applied to. The return value of the filter function is expected to be a logical vector that can be used to subset `x` to include those elements of `x` satisfying the filter.

`name`: Object of class `"ScalarCharacter"` representing the name of the filter. The name is useful for suggesting the purpose of the filter, and for debugging failed filters.

Extends

Class `"function"`, from data part. Class `".SRUtil"`, directly. Class `"OptionalFunction"`, by class "function", distance 2. Class `"PossibleMethod"`, by class "function", distance 2.

Methods

srFilter `signature(fun = "SRFilter")`: Return the function representing the underlying filter; this is primarily for interactive use to understanding filter function; usually the filter is invoked as a normal function call, as illustrated below

name `signature(x = "SRFilter")`: Return, as a `ScalarCharacter`, the name of the function.

show `signature(object = "SRFilter")`: display a brief summary of the filter

coerce `signature(from = "SRFilter", to = "FilterRules")`: Coerce a filter to a `FilterRules` object of length one.

c `signature(x = "SRFilter", ...)`: Combine filters into a single `FilterRules` object.

Author(s)

Martin Morgan <mtmorgan@fhcrc.org>

See Also

`srFilter` for predefined and user-defined filters.

Examples

```
## see ?srFilter
```

SRFilterResult-class *"SRFilterResult" for SRFilter output and statistics*

Description

Objects of this class are logical vectors indicating records passing the applied filter, with an associated data frame summarizing the name, input number of records, records passing filter, and logical operation used for all filters in which the result participated.

Usage

```
SRFilterResult(x = logical(), name = NA_character_,
   input = length(x), passing = sum(x), op = NA_character_)
## S4 method for signature 'SRFilterResult,SRFilterResult'
Logic(e1, e2)
## S4 method for signature 'SRFilterResult'
name(x, ...)
stats(x, ...)
## S4 method for signature 'SRFilterResult'
show(object)
```

Arguments

x, object, e1, e2	For SRFilterResult, logical() indicating records that passed filter or, for others, an instance of SRFilterResult class.
name	character() indicating the name by which the filter is to be referred. Internally, name, input, passing, and op may all be vectors representing columns of a data.frame summarizing the application of successive filters.
input	integer() indicating the length of the original input.
passing	integer() indicating the number of records passing the filter.
op	character() indicating the logical operation, if any, associated with this filter.
...	Additional arguments, unused in methods documented on this page.

Objects from the Class

Objects can be created through SRFilterResult, but these are automatically created by the application of srFilter instances.

Slots

.Data: Object of class "logical" indicating records that passed the filter.

name: Object of class "ScalarCharacter" representing the name of the filter whose results are summarized. The name is either the actual name of the filter, or a combination of filter names and logical operations when the outcome results from application of several filters in a single logical expression.

stats: Object of class "data.frame" summarizing the name, input number of records, records passing filter, and logical operation used for all filters in which the result participated. The data.frame rows correspond either to single filters, or to logical combinations of filters.

Extends

Class "logical", from data part. Class ".SRUtil", directly. Class "vector", by class "logical", distance 2. Class "atomic", by class "logical", distance 2. Class "vectorORfactor", by class "logical", distance 3.

Methods

Logic `signature(e1 = "SRFilterResult", e2 = "SRFilterResult")`: logic operations on filters.

! `signature(x = "SRFilterResult")`: Negate the outcome of the current filter results

name `signature(x = "SRFilterResult")`: The name of the filter that the results are based on.

stats `signature(x = "SRFilterResult")`: a `data.frame` as described in the 'Slots' section of this page.

show `signature(object = "SRFilterResult")`: summary of filter results.

Author(s)

Martin Morgan mailto:mtmorgan@fhcrc.org

See Also

srFilter

Examples

```
fa <- srFilter(function(x) x %% 2 == 0, "Even")
fb <- srFilter(function(x) x %% 2 == 1, "Odd")

x <- 1:10
fa(x) | fb(x)
fa(x) & fb(x)
!(fa(x) & fb(x))
```

SRSet-class *(Legacy) A base class for Roche experiment-wide data*

Description

This class coordinates phenotype (sample) and sequence data, primarily as used on the Roche platform.

Conceptually, this class has reads from a single experiment represented as a long vector, ordered by sample. The `readCount` slot indicates the number of reads in each sample, so that the sum of `readCount` is the total number of reads in the experiment. The `readIndex` field is a light-weight indicator of which reads from all those available that are currently referenced by the `SRSet`.

Objects from the Class

Objects of this class are not usually created directly, but instead are created by a derived class, e.g., RocheSet.

Slots

`sourcePath`: Object of class `"ExperimentPath"`, containing the directory path where sequence files can be found.

`readIndex`: Object of class `"integer"` indicating specific sequences included in the experiment.

`readCount`: Object of class `"integer"` containing the number of reads in each sample included in the experiment. The sum of this vector is the total number of reads.

`phenoData`: Object of class `"AnnotatedDataFrame"` describing each sample in the experiment. The number of rows of `phenoData` equals the number of elements in `readCount`.

`readData`: Object of class `"AnnotatedDataFrame"` containing annotations on all reads.

Extends

Class `".ShortReadBase"`, directly.

Methods

experimentPath `signature(object = "SRSet")`: return the `ExperimentPath` associated with this object.

phenoData `signature(object = "SRSet")`: return the `phenoData` associated with this object.

readCount `signature(object="SRSet")`:

readIndex `signature(object="SRSet")`:

readData `signature(object="SRSet")`:

sourcePath `signature(object="SRSet")`: Retrieve the corresponding slot from `object`.

show `signature(object = "SRSet")`: display the contents of this object.

detail `signature(x = "SRSet")`: provide more extensive information on the object.

Author(s)

Michael Lawrence <mflawrence@fhcrc.org>

Examples

```
showClass("SRSet")
```

`SRUtil-class` *".SRUtil" and related classes*

Description

These classes provide important utility functions in the **ShortRead** package, but may occasionally be seen by the user and are documented here for that reason.

Objects from the Class

Utility classes include:

- `.SRUtil-class` a virtual base class from which all utility classes are derived.
- `SRError-class` created when errors occur in **ShortRead** package code.
- `SRWarn-class` created when warnings occur in **ShortRead** package code
- `SRList-class` representing a list (heterogeneous collection) of objects. The S4Vectors::SimpleList class is a better choice for a list-like container.
- `SRVector-class` representing a vector (homogeneous collection, i.e., all elements of the same class) of objects.

Objects from these classes are not normally constructed by the user. However, constructors are available, as follows.

`SRError(type, fmt, ...)`, `SRWarn(type, fmt, ...)`:

type `character(1)` vector describing the type of the error. `type` must come from a pre-defined list of types.

fmt a `sprintf`-style format string for the message to be reported with the error.

... additional arguments to be interpolated into `fmt`.

`SRList(...)`

... elements of any type or length to be placed into the `SRList`. If the length of `...` is 1 and the argument is a list, then the list itself is placed into `SRList`.

`SRVector(..., vclass)`

... elements all satisfying an `is` relationship with `vclass`, to be placed in `SRVector`.

vclass the class to which all elements in `...` belong. If `vclass` is missing and `length(list(...))` is greater than zero, then `vclass` is taken to be the class of the first argument of `...`.

`SRVector` errors:

SRVectorClassDisagreement this error occurs when not all arguments `...` satisfy an 'is' relationship with `vclass`.

Slots

`SRError` and `SRWarn` have the following slots defined:

`.type`**:** Object of class `"character"` containing the type of error or warning. `.type` must come from a pre-defined list of types, see, e.g., `ShortRead:::.SRError_types`.

`.message`**:** Object of class `"character"` containing a detailed message describing the error or warning.

`SRList` has the following slot defined:

`.srlist`**:** Object of class `"list"` containing the elements in the list.

`SRVector` extends `SRList`, with the following additional slot:

`vclass`**:** Object of class `"character"` naming the type of object all elements of `SRVector` must be.

Methods

Accessors are available for all slots, and have the same name as the slot, e.g., `vclass` to access the `vclass` slot of `SRVector`. Internal slots (those starting with '.' also have accessors, but these are not exported e.g., `ShortRead:::.type`.

`SRList` has the following methods:

length `signature(x = "SRList")`: return the (`integer(1)`) length of the `SRList`.

names `signature(x = "SRList")`: return a character vector of list element names. The length of the returned vector is the same as the length of `x`.

names<- `signature(x = "SRList", value = "character")`: assign `value` as names for members of `x`.

[`signature(x = "SRList", i = "ANY", j = "missing")`: subset the list using standard R list subset paradigms.

[[`signature(x = "SRList", i = "ANY", j = "missing")`: select element 'i' from the list, using standard R list selection paradigms.

lapply `signature(X = "SRList", FUN="ANY")`: apply a function to all elements of `X`, with additional arguments interpreted as with `lapply`.

sapply `signature(X = "SRList")`: apply a function to all elements of `X`, simplifying the result if possible. Additional arguments interpreted as with `sapply`.

srlist `signature(object="SRList")`: coerce the SRList object to a list.

show `signature(object = "SRList")`: display an informative summary of the object content, including the length of the list represented by `object`.

detail `signature(x = "SRList")`: display a more extensive version of the object, as one might expect from printing a standard list in R.

`SRVector` inherits all methods from `SRList`, and has the following additional methods:

show `signature(object = "SRVector")`: display an informative summary of the object content, e.g., the vector class (`vclass`) and length.

detail `signature(x = "SRVector")`: display a more extensive version of the object, as one might expect from a printing a standard R list.

Author(s)

Martin Morgan

Examples

```
getClass(".SRUtil", where=getNamespace("ShortRead"))
ShortRead:::.SRError_types
ShortRead:::.SRWarn_types

detail(SRList(1:5, letters[1:5]))

tryCatch(SRVector(1:5, letters[1:5]),
         SRVectorClassDisagreement=function(err) {
             cat("caught:", conditionMessage(err), "\n")
         })
```

tables *Summarize XStringSet read frequencies*

Description

This generic summarizes the number of times each sequence occurs in an XStringSet instance.

Usage

```
tables(x, n=50, ...)
```

Arguments

x	An object for which a tables method is defined.
n	An integer(1) value determining how many named sequences will be present in the top portion of the return value.
...	Additional arguments available to methods

Details

Methods of this generic summarize the frequency with which each read occurs, There are two components to the summary. The reads are reported from most common to least common; typically a method parameter controls how many reads to report. Methods also return a pair of vectors describing how many reads were represented 1, 2, ... times.

The following methods are defined, in addition to methods described in class-specific documentation:

tables `signature(x="XStringSet", n=50)`: Apply `tables` to the `XStringSet` `x`.

Value

A list of length two.

top	A named integer vector. Names correspond to sequences. Values are the number of times the corresponding sequence occurs in the XStringSet. The vector is sorted in decreasing order; methods typically include a parameter specifying the number of sequences to return.
distribution	a data.frame with two columns. nOccurrences is the number of times any particular sequence is represented in the set (1, 2, ...). nReads is the number of reads with the corresponding occurrence.

Author(s)

Martin Morgan <mtmorgan@fhcrc.org>

Examples

```
showMethods("tables")
sp <- SolexaPath(system.file("extdata", package="ShortRead"))
aln <- readAligned(sp)
tables(sread(aln), n=6)
lattice::xyplot(log10(nReads)~log10(nOccurrences),
       tables(sread(aln))$distribution)
```

`trimTails` *Trim ends of reads based on nucleotides or qualities*

Description

These generic functions remove leading or trailing nucleotides or qualities. `trimTails` and `trimTailw` remove low-quality reads from the right end using a sliding window (`trimTailw`) or a tally of (successive) nucleotides falling at or below a quality threshold (`trimTails`). `trimEnds` takes an alphabet of characters to remove from either left or right end.

Usage

```
## S4 methods for 'ShortReadQ', 'FastqQuality', or 'SFastqQuality'
trimTailw(object, k, a, halfwidth, ..., ranges=FALSE)
trimTails(object, k, a, successive=FALSE, ..., ranges=FALSE)
trimEnds(object, a, left=TRUE, right=TRUE, relation=c("<=", "=="),
    ..., ranges=FALSE)

## S4 method for signature 'BStringSet'
trimTailw(object, k, a, halfwidth, ..., alphabet, ranges=FALSE)
## S4 method for signature 'BStringSet'
trimTails(object, k, a, successive=FALSE, ...,
    alphabet, ranges=FALSE)

## S4 method for signature 'character'
trimTailw(object, k, a, halfwidth, ..., destinations, ranges=FALSE)
## S4 method for signature 'character'
trimTails(object, k, a, successive=FALSE, ..., destinations, ranges=FALSE)
## S4 method for signature 'character'
trimEnds(object, a, left=TRUE, right=TRUE, relation=c("<=", "=="),
    ..., destinations, ranges=FALSE)
```

Arguments

`object`	An object (e.g., `ShortReadQ` and derived classes; see below to discover these methods) or character vector of fastq file(s) to be trimmed.
`k`	`integer(1)` describing the number of failing letters required to trigger trimming.

a	For trimTails and trimTailw, a character(1) with nchar(a) == 1L giving the letter at or below which a nucleotide is marked as failing. For trimEnds a character() with all nchar() == 1L giving the letter at or below which a nucleotide or quality scores marked for removal.
halfwidth	The half width (cycles before or after the current; e.g., a half-width of 5 would span 5 + 1 + 5 cycles) in which qualities are assessed.
successive	logical(1) indicating whether failures can occur anywhere in the sequence, or must be successive. If successive=FALSE, then the k'th failed letter and subsequent are removed. If successive=TRUE, the first succession of k failed and subsequent letters are removed.
left, right	logical(1) indicating whether trimming is from the left or right ends.
relation	character(1) selected from the argument values, i.e., "<=" or "==" indicating whether all letters at or below the alphabet(object) are to be removed, or only exact matches.
...	Additional arguments, perhaps used by methods.
destinations	For object of type character(), an equal-length vector of destination files. Files must not already exist.
alphabet	character() (ordered low to high) letters on which quality scale is measured. Usually supplied internally (user does not need to specify). If missing, then set to ASCII characters 0-127.
ranges	logical(1) indicating whether the trimmed object, or only the ranges satisfying the trimming condition, be returned.

Details

trimTailw starts at the left-most nucleotide, tabulating the number of cycles in a window of 2 * halfwidth + 1 surrounding the current nucleotide with quality scores that fall at or below a. The read is trimmed at the first nucleotide for which this number >= k. The quality of the first or last nucleotide is used to represent portions of the window that extend beyond the sequence.

trimTails starts at the left-most nucleotide and accumulates cycles for which the quality score is at or below a. The read is trimmed at the first location where this number >= k. With successive=TRUE, failing qualities must occur in strict succession.

trimEnds examines the left, right, or both ends of object, marking for removal letters that correspond to a and relation. The trimEnds,ShortReadQ-method trims based on quality.

ShortReadQ methods operate on quality scores; use sread() and the ranges argument to trim based on nucleotide (see examples).

character methods transform one or several fastq files to new fastq files, applying trim operations based on quality scores; use filterFastq with your own filter argument to filter on nucleotides.

Value

An instance of class(object) trimmed to contain only those nucleotides satisfying the trim criterion or, if ranges=TRUE an IRanges instance defining the ranges that would trim object.

Note

The `trim*` functions use OpenMP threads (when available) during creation of the return value. This may sometimes create problems when a process is already running on multiple threads, e.g., with an error message like

```
libgomp: Thread creation failed: Resource temporarily unavailable
```

A solution is to precede problematic code with the following code snippet, to disable threading

```
nthreads <- .Call(ShortRead:::.set_omp_threads, 1L)
on.exit(.Call(ShortRead:::.set_omp_threads, nthreads))
```

Author(s)

Martin Morgan <mtmorgan@fhcrc.org>

Examples

```
showMethods(trimTails)

sp <- SolexaPath(system.file('extdata', package='ShortRead'))
rfq <- readFastq(analysisPath(sp), pattern="s_1_sequence.txt")

## remove leading / trailing quality scores <= 'I'
trimEnds(rfq, "I")
## remove leading / trailing 'N's
rng <- trimEnds(sread(rfq), "N", relation="==", ranges=TRUE)
narrow(rfq, start(rng), end(rng))
## remove leading / trailing 'G's or 'C's
trimEnds(rfq, c("G", "C"), relation="==")
```

Utilites *Utilities for common, simple operations*

Description

These functions perform a variety of simple operations.

Usage

```
polyn(nucleotides, n)
```

Arguments

`nucleotides`	A character vector with all elements having exactly 1 character, typically from the IUPAC alphabet.
`n`	An `integer(1)` vector.

Details

polyn returns a character vector with each element having n characters. Each element contains a single nucleotide. Thus polyn("A", 5) returns AAAAA.

Value

polyn returns a character vector of length length(nucleotide)

Author(s)

Martin Morgan <mtmorgan@fhcrc.org>

Examples

```
polyn(c("A", "N"), 35)
```

Index