

ShortRead

April 19, 2009

AlignedDataFrame-class

"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](#).

Author(s)

Martin Morgan <mtmorgan@fhcrc.org>

See Also

[AnnotatedDataFrame](#)

AlignedDataFrame *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>

AlignedRead-class *"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.
- 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 "DNAStrngSet" 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.

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

coverage signature(x = "AlignedRead", start = NA, end = NA, ..., coords=c("leftmost", "fiveprime"), extend=0L):

Calculate coverage across reads present in `x`.

`start` and `end` are regions (e.g., of chromosomes) over which coverage is to be calculated. If provided, these are length 1 integers or *named* integer vectors of length greater than 1. If named integer vectors, the names must match `levels(chromosome(x))`. If omitted, coverage is calculated over the range of values spanned by the reads in `x`

`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.

srorder \signature{x = "AlignedRead"}:

srrank \signature{x = "AlignedRead"}:

srsort \signature{x = "AlignedRead"}:

sruplicated \signature{x = "AlignedRead"}: Order, rank, sort, and find duplicates in `AlignedRead` objects. Reads are sorted by `chromosome`, `strand`, `position`, and then `sread`; less fine-grained sorting can be accomplished with, e.g., `x[srorder(sread(x))]`.

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

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

Author(s)

Martin Morgan <mtmorgan@fhcrc.org>

See Also

[readAligned](#)

Examples

```
showMethods(class="AlignedRead")
dirPath <- system.file('extdata', 'maq', package='ShortRead')
readAligned(dirPath, 'out.aln.1.txt', type="MAQMapView")
```

AlignedRead	<i>Construct objects of class "AlignedRead"</i>
-------------	---

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 <code>DNAStrngSet</code> , containing the DNA sequences of the short reads.
id	An object of class <code>BStringSet</code> , containing the identifiers of the short reads. This object is the same length as <code>sread</code> .
quality	An object of class <code>BStringSet</code> , containing the ASCII-encoded quality scores of the short reads. This object is the same length as <code>sread</code> .
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 <code>AlignedDataFrame</code> with number of rows equal to the length of <code>sread</code> , containing additional information about alignments.

Value

An object of class [AlignedRead](#).

Author(s)

Martin Morgan <mtmorgan@fhcrc.org>

See Also

[AlignedRead](#).

ExperimentPath-class

"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(object = "ExperimentPath")`: summarize file paths of object.

Author(s)

Michael Lawrence

Examples

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

`.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

There are no methods defined directly on the QA class; see derived class help pages for additional methods.

Author(s)

Martin Morgan <mtmmorgan@fhcrc.org>

See Also

[SolexaExportQA](#).

Examples

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

`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.

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.

length signature(x = "QualityScore"):

length signature(x = "MatrixQuality"):

The numeric(1) length (e.g., number of reads) represented by the quality score.

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(object = "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.

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"):

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

srsort signature(x = "FastqQuality"):

srorder signature(x = "FastqQuality"):

srrank signature(x = "FastqQuality"):

sruplicated signature(x = "FastqQuality"):

Apply [srsort](#), [srorder](#), [srrank](#), and [sruplicated](#) to quality scores, returning objects as described on the appropriate help page.

Author(s)

Martin Morgan <mtmmorgan@fhcrc.org>

See Also

[NumericQuality](#) and other constructors.

Examples

```
names(slot(getClass("QualityScore"), "subclasses"))
```

QualityScore	<i>Construct objects indicating read or alignment quality</i>
--------------	---

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])
```

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

Description

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

Objects from the Class

Objects from the class are created with the `RochePath` constructor:

```
RochePath(experimentPath = NA_character_, readPath = .srPath(experimentPath,
"run"), 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 = "character"`, `pattern=".\\.fna$"`, `sample = 1`, ...):

readFasta signature(`dirPath = "RochePath"`, `pattern=".\\.fna$"`, `sample = 1`, `run`, ...):

Read sequences from files matching `list.files(dirPath, pattern)` (when `dirPath="character"`) or `list.files(readPath(dir)[run], pattern)`, retaining reads corresponding to `sample`. The result is a `DNAStrngSet`.

readQual signature(`dirPath = "RochePath"`, `pattern=".\\.qual$"`, `reads=NULL`, `sample=1`, `run`, ...):

Read quality scores from files matching `list.files(qualPath(dirPath)[run])`, corresponding to `sample`. Non-null `reads` is used as an (optional) template for parsing quality scores.

read454 signature(`dirPath = "RochePath"`): read sequences and quality scores into a [ShortReadQ](#).

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(`object = "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	<i>Roche (454) experiment-wide data container</i>
----------------	---

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")
```

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

Author(s)

Martin Morgan <mtmorgan@fhcrc.org>

See Also

`srFilter` for predefined and user-defined filters.

Examples

```
## see ?srFilter
```

SRSet-class

*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 lightweight 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.

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 (object = "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.
- `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"`): 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`.
- show** signature(`object = "SRList"`): display an informative summary of the object content, including the length of the list represented by `object`.
- detail** signature(`object = "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(`object = "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")
         })

```

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 not normally created directly by the user.

Slots

sread: Object of class "DNAStrngSet" 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

See [accessors](#) for slot accessor functions.

[[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.

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

width [signature\(x = "ShortRead"\)](#): returns a `integer(1)` vector describing the number of characters in each read in this object.

srorder [\signature{x = "ShortRead"}](#):

srrank [\signature{x = "ShortRead"}](#):

srsort [\signature{x = "ShortRead"}](#):

- sruplicated** \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.
- 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(object = "ShortRead"): provides a more extensive summary of this object, displaying the first and last entries of `sread` and `id`.

Author(s)

Martin Morgan

See Also

[ShortReadQ](#)

Examples

```
showClass("ShortRead")
showMethods(class="ShortRead")
```

ShortReadBase-package

Base classes and methods for high-throughput short-read sequencing data.

Description

Base classes, functions, and methods for representation of high-throughput, short-read sequencing data.

Details

See `packageDescription('ShortRead')`

Author(s)

Maintainer: Martin Morgan <mtmorgan@fhcrc.org>

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 not normally created directly by the user, but are instead the result of [readFastq](#).

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

See [accessors](#) for additional functions to access slot content, and [ShortRead](#) for inherited methods. Additional methods include:

[`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.

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.

detail `signature(object = "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", inherit=FALSE)
showMethods(class="ShortRead", inherit=FALSE)
```

SolexaExportQA-class

Quality assessment summaries from Solexa export files

Description

This class contains a list-like structure with summary descriptions derived from visiting one or more Solexa 'export' 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 "[SRLList](#)", directly. Class "[.QA](#)", directly. Class "[.SRUtil](#)", by class "SRLList", distance 2. Class "[.ShortReadBase](#)", by class ".QA", distance 2.

Methods

Accessor methods are inherited from the [SRLList](#) class.

A additional methods defined on this class are:

report signature (x="SolexaExportQA", ..., dest=paste(tempfile(), "pdf", sep="."), type="pdf"): produces a pdf file summarizing the QA results.

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

Author(s)

Martin Morgan <mtmorgan@fhcrc.org>

See Also

[qa](#).

Examples

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

SolexaPath-class *"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(dataPath, "GoldCrest"), imageAnalysisPath=.solexaPath(dataPath,
"^C"), baseCallPath=.solexaPath(imageAnalysisPath, "^Bustard"), analysisPath=.solexaPath(
"^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.

experimentPath See 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:

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.

readFastq signature(`dirPath = "SolexaPath"`, `pattern = ".*_sequence.txt"`, `run, ...`):

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. Note that the default method reads *all* sequence files into a single object; often one will want to specify a pattern for each lane.

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.

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 files, returning a [SolexaExportQA](#) object summarizing quality assessment. If `Rmpi` has been initiated, quality assessment calculations are distributed across available nodes (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(`object = "SolexaPath"`): summarize file paths of `object`. All file paths are presented in full.

Author(s)

Martin Morgan

Examples

```

showClass("SolexaPath")
showMethods(class="SolexaPath")
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)

```

SolexaSet-class	<i>"SolexaSet" coordinating Solexa output locations with sample annotations</i>
-----------------	---

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(object = "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"` 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")
## 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)
```

Description

These functions are 'accessors' (to get and set values) for objects in the **ShortRead** package.

Usage

```
## SRVector
vclass(object, ...)
## ShortRead / ShortReadQ
sread(object, ...)
id(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

<code>object</code>	An object derived from class <code>ShortRead</code> . 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 <code>sread</code>) from <code>object</code> .
<code>...</code>	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))
```

alphabetByCycle *Summarize short read nucleotide or quality scores by cycle*

Description

alphabetByCycle summarizes short read nucleotides 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 or quality scores to be summarized. All entries in the string set must have the same width (i.e., number of characters in each read or quality score).
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 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	<i>Efficiently calculate the sum of quality scores across bases</i>
---------------	---

Description

This generic takes a [QualityScore](#) 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>

clean	<i>Remove sequences with ambiguous nucleotides from short read classes</i>
-------	--

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 containin 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	<i>Count lines in all (text) files in a directory whose file name matches a pattern</i>
------------	---

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.

Usage

```
basePath(object, ...)
```

Arguments

<code>object</code>	For <code>basePath</code> , and object of class <code>ExperimentPath</code> .
<code>...</code>	Additional arguments.

Author(s)

Martin Morgan

detail

Show (display) detailed object content

Description

This is a variant of `show`, offering a more detailed display of object content.

Usage

```
detail(object, ...)
```

Arguments

`object` An object derived from class `ShortRead`. See help pages for individual objects, e.g., [ShortReadQ](#). The default simply invokes `show`.

`...` Additional arguments. The default definition makes no use of these arguments.

Value

None; the function is invoked for its side effect (detailed display of object content).

Author(s)

Martin Morgan

Examples

```
sp <- SolexaPath(system.file('extdata', package='ShortRead'))
show(sp)
detail(sp)
```

pileup

Calculate a pile-up representation of short-read mappings

Description

Given short read mappings or similar data, this function calculates a pile-up, i.e. representing the reference sequence (that is, typically, one of the chromosome), such that its length is the number of base pairs of the reference sequence, and each integer is the number of reads (or fragments, see below) mapped to the corresponding basepair.

Usage

```
pileup( start, fraglength, chrlength,
        dir = factor( "+", levels=c("-", "+") ),
        readlength = fraglength,
        offset = 1 )
```

Arguments

`start` A vector with the start positions of each read on the reference sequence. All reads must correspond to the same reference sequence.

`fraglength` A vector of the same length as `'start'` with the lengths of all the fragments. Alternatively, a single integer, specifying one constant length to assume for all tags.

`chrlength` The length of the reference sequence. You may use the function `readBfaToc` to extract this information from the `.bfa` file.

`dir` A factor with level `"-"` and `"+"` of the same length as `'start'`, specifying whether the fragment extends to the right (towards higher index values, `'+'`) or to the left (towards lower index values, `'-'`) beyond the read. See below for more explanation.

readlength	The length of the reads, either as a vector of the same length as 'start' or as a single number. This parameter makes sense only if 'dir' is used, too. If not specified, read lengths and fragment lengths are taken to be the same.
offset	The index of the first base pair in the result vector. The default is 1, i.e. assumes that the 'start' positions are in 1-based chromosome coordinates.

Value

an integer vector of length 'chrlength', each element counting how many fragments map to this basepair.

Note

1. This function is not (yet) suitable for paired-end reads.
2. If the arguments 'dir' and 'readlength' are not used, the fragments are assumed to start at the positions given in 'start' and extend to the right by the number of basepairs given in fraglength. If 'dir' and 'readlength' are supplied then the interval starting at 'start' and extending to the right by the number of base pairs given in 'readlength' marks the position of the read, which is one end of the fragment. If 'dir' is '+', it is taken as the left end and the fragment will be extended to the right to have the total length given by 'fraglength'. If 'dir' is '-', the end is taken as the right end and is extended to the left. Note that in the latter case, the 'start' position does mark the border between read and rest of fragment, not an actual 'end' of the fragment. If you are confused now, look at the examples below.
3. Sorry for the inconsequent use of 'width' and 'length' in a seemingly interchangeable fashion.

Author(s)

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

Examples

```
## Not run:
```

Example 1: Assuming that 'lane' is a AlignedRead object containing aligned reads from a Solexa lane, you may get a pile-up representation of chromosome 13 as follows

```
chr13length <- 114142980 # the length of human chromosome 13
pu <- pileup( position(lane)[chromosome(lane)=="13"], width(lane), chr13length )
```

Example 2: Even though the width of the reads (as reported by `width(lane)`) is only these 24 bp are just one end of a longer fragment. Assuming that all fragments have been sonicated to about the same length, say 150 bp, we may get a better pile-up representation

```
pu2 <- pileup( position(lane)[chromosome(lane)=="13"], 150, chr13length,
strand(lane)[chromosome(lane)=="13"], width(lane) )
```

```
## End(Not run)
```

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)
```

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, ...)
```

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.

`...` Additional arguments used by methods.

Details

The following methods are defined:

`type="SolexaExport"`, ... 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.

Value

An object derived from class `.QA`

Author(s)

Martin Morgan <mtmorgan@fhcrc.org>

See Also

`.QA`, `SolexaExportQA`

Examples

```
showMethods("qa")
```

`readAligned` *Read aligned reads and their quality scores into R representations*

Description

`readAligned` reads all aligned read files in a directory `dirPath` whose file name matches `pattern`, returning a compact internal representation of the alignments, 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 aligned read file.

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) of aligned read files to be input.
pattern	The (grep -style) pattern describing file names to be read. The default (<code>character(0)</code>) results in (attempted) input of all files in the directory.
...	Additional arguments, used by methods. Most methods implement <code>filter=srFilter()</code> , 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 Ignored
Run number stored in `alignData`
Lane stored in `alignData`
Tile stored in `alignData`
X stored in `alignData`
Y stored in `alignData`
Index string Ignored
Read number Ignored
Read `sread`
Quality `quality`
Match chromosome `chromosome`
Match contig Ignored
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`

Paired read columns are not interpreted. The resulting [AlignedRead](#) object does *not* contain a meaningful `id`; instead, use information from `alignData` to identify reads.

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="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.

`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.)

`type="MAQMapview"` Parse alignment files created by MAQ's 'mapview' 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.

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

A `AlignedRead` object.

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

Examples

```
sp <- SolexaPath(system.file("extdata", package="ShortRead"))
ap <- analysisPath(sp)
## ELAND_EXTENDED
readAligned(ap, "s_2_export.txt", "SolexaExport")
## PhageAlign
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
readAligned(dirPath, type="MAQMapview")

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

readBaseQuality *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, seqPattern=character(0), prbPattern=character(0), ...)
```

Arguments

<code>dirPath</code>	A character vector (or other object; see methods defined on this generic) giving the directory path (relative or absolute) of files to be input.
<code>seqPattern</code>	The (grep-style) pattern describing base call file names to be read. The default (<code>character(0)</code>) results in (attempted) input of all files in the directory.
<code>prbPattern</code>	The (grep-style) pattern describing quality score file names to be read. The default (<code>character(0)</code>) results in (attempted) input of all files in the directory.
<code>...</code>	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`*Get a list of the sequences in a Maq .bfa file*

Description

As `pileup` 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.

`readFastq` *Read FASTQ-formatted files into compact R representations*

Description

`readFastq` reads all FASTQ-formatted 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.

Usage

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

Arguments

<code>dirPath</code>	A character vector (or other object; see methods defined on this generic) giving the directory path (relative or absolute) of FASTQ files to be read.
<code>pattern</code>	The (grep-style) pattern describing file names to be read. The default (<code>character(0)</code>) results in line (attempted) input of all files in the directory.
<code>...</code>	Additional arguments, perhaps used by methods.

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.

Value

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.

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("readFastq")

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
```

readPrb

Read Solexa prb files as fastq-style quality scores

Description

readPrb reads all `_prb.txt` files in a directory into a single object of class `SFastqQuality`. Methods do this by identifying the maximum base call quality for each cycle and read, and representing this as an ASCII-encoded character string following Solexa conventions.

Usage

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

Arguments

dirPath	Directory path or other object (e.g., <code>SolexaPath</code> for which methods are defined).
pattern	Regular expression matching names of <code>_prb</code> files to be summarized.
...	Additional arguments, e.g., to <code>srapply</code> , used during evaluation.

Value

An object of class `SFastqQuality`

Author(s)

Martin Morgan <mtmorgan@fhcrc.org>

See Also

~~objects to See Also as [help](#), ~~~

Examples

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

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), 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 (<code>character(0)</code>) reads all files in <code>dirPath</code> . 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 <code>colClasses</code> equal to <code>NULL</code> are ignored. Other entries in <code>colClasses</code> 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.
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", "DNAStrng", "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)
```

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 = "pdf")
```

Arguments

<code>x</code>	An object returned by <code>qa</code> , usually derived from class <code>.QA</code>
<code>...</code>	Additional arguments used by specific methods. See specific methods for details.
<code>dest</code>	The output destination for the final report; destination files are overwritten if they already exist.
<code>type</code>	A text string defining the type of report; available report types depend on the type of object <code>x</code> .

Details

The following methods are defined:

`tempfile(), type="pdf"` 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")
```

srFilter

Functions for user-created and built-in ShortRead filters

Description

These functions create user-defined (`srFilter`) 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(filter, ..., .name)

chromosomeFilter(regex=character(0), .name="ChromosomeFilter")
strandFilter(strandLevels=character(0), .name="StrandFilter")
nFilter(threshold=0L, .name="CleanNFilter")
polynFilter(threshold=0L, nuc=c("A", "C", "T", "G", "other"),
            .name="PolyNFilter")
srdistanceFilter(subject=character(0), threshold=0L,
                .name="SRDistanceFilter")
alignQualityFilter(threshold=0L, .name="AlignQualityFilter")
alignDataFilter(expr=expression(), .name="AlignDataFilter")
```

Arguments

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

regex	Either character(0) or character(1) regular expression used as <code>grep(regex, chromosome(x))</code> to filter based on chromosome. The default (<code>character(0)</code>) performs no filtering
strandLevels	Either character(0) or character(1) containing strand levels to be selected. <code>ShortRead</code> objects have standard strand levels <code>NA</code> , <code>"+"</code> , <code>"-"</code> , <code>"*"</code> , with <code>NA</code> meaning strand information not available and <code>"*"</code> meaning strand information not relevant.
threshold	A numeric(1) value representing a minimum (<code>srDistanceFilter</code> , <code>alignQualityFilter</code>) or maximum (<code>nFilter</code> , <code>polynFilter</code>) criterion for the filter. The minima and maxima are closed-interval (i.e., $x \geq \text{threshold}$, $x \leq \text{threshold}$ for some property x of the object being filtered).
nuc	A character vector containing IUPAC symbols for nucleotides or the value <code>"other"</code> corresponding to all non-nucleotide symbols, e.g., <code>N</code> .
subject	A character() of any length, to be used as the corresponding argument to <code>srDistance</code> .
expr	A expression to be evaluated with <code>pData(alignData(x))</code> .
...	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.

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

`strandFilter` selects elements satisfying `match(strand(x), strand, nomatch=0) > 0`.

`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`.

`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
chr5xy <- compose(filt, xy)
aln[chr5xy(aln)]

# custom filter: minimum calibrated base call quality >20
goodq <- srFilter(function(x) {
  apply(as(quality(x), "matrix"), 1, min) > 20
}, name="GoodQualityBases")
goodq
aln[goodq(aln)]
```

 srapply

Apply-like function for distribution across MPI-based clusters.

Description

This `lapply` like function evaluates locally or, if **Rmpi** is loaded and workers spawned, across nodes in a cluster. Errors in evaluation of `FUN` generate warnings; results are trimmed to exclude results where the error occurs.

Usage

```
srapply(X, FUN, ..., fapply = .fapply(), reduce = .reduce(), verbose = FALSE)
```

Arguments

`X` Tasks to be distributed. `X` should be an object for which `lapply` or `sapply` are defined (more precisely, `mpi.parLapply`, `mpi.parSapply`). Performance is best when these objects are relatively small, e.g., file names, compared to the work to be done on each by `FUN`.

<code>FUN</code>	A function to be applied to each element of <code>X</code> . The function must have <code>...</code> or named argument <code>verbose</code> in its signature. It is best if it makes no reference to variables other than those in its argument list. or in loaded packages (the ShortRead package is available on remote nodes).
<code>...</code>	Additional arguments, passed to <code>FUN</code> .
<code>fapply</code>	An optional argument defining an <code>lapply</code> -like function to be used in partitioning <code>X</code> . See details, below.
<code>reduce</code>	Optional function accepting a list (the result of <code>fapply</code> and summarizing this. The default reports errors in function evaluation as warnings, returning the remaining values as elements of a list. See details below for additional hints.
<code>verbose</code>	Report whether evaluation is local or <code>mpi</code> -based; also forwarded to <code>FUN</code> , allowing detailed reports from remote instances.

Details

The default value for `fapply` is available with `ShortRead:::fapply()`. It tests whether **Rmpi** is loaded and workers spawned. If so, the default ensures that **ShortRead** is required on all workers, and then invokes `mpi.parLapply` with arguments `X`, `FUN`, `...`, and `verbose`. The function `FUN` is wrapped so that errors are returned as objects of class `SRError` with type `RemoteError`.

If no workers are available, the code evaluates `FUN` so that errors are reported as with remote evaluation.

Custom `reduce` functions might be written as `reduce=function(lst) unlist(lst, use.names=TRUE)`.

Value

The returned value depends on the value of `reduce`, but by default is a list with elements containing the results of `FUN` applied to each of `X`. Evaluations resulting in an error have been removed, and a warning generated.

Author(s)

Martin Morgan <mtmorgan@fhcrc.org>

Examples

```
## ... or 'verbose' required in argument,
srapply(1:10, function(i, ...) i)
## collapse result to vector
srapply(1:10, function(i, ...) i, reduce=unlist)
x <- srapply(1:10, function(i, ...) {
  if (runif(1)<.2) stop("oops") else i
})
length(x) ## trimmed to exclude errors
```

srdistance	<i>Edit distances between reads and a small number of short references</i>
------------	--

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

<code>pattern</code>	An object of class <code>DNAStrngSet</code> containing reads whose edit distance is desired.
<code>subject</code>	A short character vector, <code>DNAStrng</code> or (small) <code>DNAStrngSet</code> to serve as reference.
<code>...</code>	additional arguments, forward to <code>srapply</code> .

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. `srrorder`, `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

```

srrorder(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"):

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

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

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

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

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

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

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

Value

The functions return the following values:

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

Author(s)

Martin Morgan <mtmorgan@fhcrc.org>

Examples

```
showMethods("srsort")
showMethods("srrorder")
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))
```

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

<code>x</code>	An object for which a <code>tables</code> method is defined.
<code>n</code>	An integer (1) value determining how many named sequences will be present in the <code>top</code> portion of the return value.
<code>...</code>	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)
xyplot(log10(nReads)~log10(nOccurrences),
       tables(sread(aln))$distribution)
```


Index

*Topic **IO**

readXStringColumns, 39

*Topic **classes**

.QA-class, 6
AlignedDataFrame-class, 1
AlignedRead-class, 2
ExperimentPath-class, 5
QualityScore-class, 6
RochePath-class, 9
RocheSet-class, 11
ShortRead-class, 16
ShortReadQ-class, 18
SolexaExportQA-class, 19
SolexaPath-class, 20
SolexaSet-class, 22
SRFilter-class, 12
SRSet-class, 13
SRUtil-class, 14

*Topic **manip**

accessors, 23
AlignedDataFrame, 2
AlignedRead, 4
alphabetByCycle, 25
alphabetScore, 26
clean, 26
countLines, 27
deprecated, 28
detail, 28
qa, 31
QualityScore, 8
readAligned, 32
readBaseQuality, 35
readFastq, 37
readPrb, 38
report, 40
srapply, 43
srdistance, 45
sruplicated, 46
srFilter, 41
tables, 47
Utilites, 31

*Topic **package**

ShortReadBase-package, 17

.QA, 19, 32, 40
.QA-class, 6
.Roche, 9, 11
.Roche-class
(ShortReadBase-package), 17
.SRUtil, 12, 19
.SRUtil-class (SRUtil-class), 14
.ShortReadBase, 3, 5–7, 9, 11, 13, 16,
18–20, 22
.ShortReadBase-class
(ShortReadBase-package), 17
.Solexa, 20, 22
.Solexa-class
(ShortReadBase-package), 17
[, AlignedRead, ANY, ANY, ANY-method
(AlignedRead-class), 2
[, AlignedRead, ANY, missing, ANY-method
(AlignedRead-class), 2
[, AlignedRead, missing, ANY, ANY-method
(AlignedRead-class), 2
[, AlignedRead, missing, missing, ANY-method
(AlignedRead-class), 2
[, MatrixQuality, ANY, missing, ANY-method
(QualityScore-class), 6
[, QualityScore, ANY, missing, ANY-method
(QualityScore-class), 6
[, SRList, ANY, missing, ANY-method
(SRUtil-class), 14
[, ShortRead, ANY, ANY, ANY-method
(ShortRead-class), 16
[, ShortRead, ANY, missing, ANY-method
(ShortRead-class), 16
[, ShortRead, missing, ANY, ANY-method
(ShortRead-class), 16
[, ShortRead, missing, missing, ANY-method
(ShortRead-class), 16
[, ShortReadQ, ANY, ANY, ANY-method
(ShortReadQ-class), 18
[, ShortReadQ, ANY, missing, ANY-method
(ShortReadQ-class), 18
[, ShortReadQ, missing, ANY, ANY-method
(ShortReadQ-class), 18
[, ShortReadQ, missing, missing, ANY-method

- (ShortReadQ-class)*, 18
- [[, MatrixQuality, ANY, missing-method
(QualityScore-class), 6
- [[, QualityScore, ANY, missing-method
(QualityScore-class), 6
- [[, SRList, ANY, missing-method
(SRUtil-class), 14
- accessors, 3, 16, 18, 23
- alignData (*accessors*), 23
- alignDataFilter (*srFilter*), 41
- AlignedDataFrame, 1, 2, 2
- AlignedDataFrame-class, 1
- AlignedRead, 1, 2, 4, 4, 5, 33–35
- AlignedRead-class, 2
- alignQuality (*accessors*), 23
- alignQualityFilter (*srFilter*), 41
- alphabet, FastqQuality-method
(QualityScore-class), 6
- alphabetByCycle, 7, 17, 18, 25
- alphabetByCycle, BStringSet-method
(alphabetByCycle), 25
- alphabetByCycle, FastqQuality-method
(QualityScore-class), 6
- alphabetByCycle, ShortRead-method
(ShortRead-class), 16
- alphabetByCycle, ShortReadQ-method
(ShortReadQ-class), 18
- alphabetFrequency, 7
- alphabetFrequency, FastqQuality-method
(QualityScore-class), 6
- alphabetScore, 7, 18, 26
- alphabetScore, FastqQuality-method
(QualityScore-class), 6
- alphabetScore, SFastqQuality-method
(QualityScore-class), 6
- alphabetScore, ShortReadQ-method
(ShortReadQ-class), 18
- analysisPath (*accessors*), 23
- AnnotatedDataFrame, 1, 2, 22
- baseCallPath (*accessors*), 23
- basePath (*deprecated*), 28
- BStringSet, 8
- chromosome (*accessors*), 23
- chromosomeFilter (*srFilter*), 41
- clean, 26
- clean, DNASTringSet-method
(clean), 26
- clean, ShortRead-method
(ShortRead-class), 16
- compose (*srFilter*), 41
- countLines, 27
- coverage, AlignedRead-method
(AlignedRead-class), 2
- dataPath (*accessors*), 23
- deprecated, 28
- detail, 28
- detail, .ShortReadBase-method
(SRUtil-class), 14
- detail, AlignedRead-method
(AlignedRead-class), 2
- detail, ExperimentPath-method
(ExperimentPath-class), 5
- detail, QualityScore-method
(QualityScore-class), 6
- detail, RochePath-method
(RochePath-class), 9
- detail, ShortRead-method
(ShortRead-class), 16
- detail, ShortReadQ-method
(ShortReadQ-class), 18
- detail, SolexaPath-method
(SolexaPath-class), 20
- detail, SolexaSet-method
(SolexaSet-class), 22
- detail, SRList-method
(SRUtil-class), 14
- detail, SRSet-method
(SRSet-class), 13
- detail, SRVector-method
(SRUtil-class), 14
- ExperimentPath, 9, 10, 13
- ExperimentPath
(ExperimentPath-class), 5
- experimentPath (*accessors*), 23
- experimentPath, SRSet-method
(SRSet-class), 13
- ExperimentPath-class, 5
- FastqQuality, 7
- FastqQuality (*QualityScore*), 8
- FastqQuality, BStringSet-method
(QualityScore), 8
- FastqQuality, character-method
(QualityScore), 8
- FastqQuality, missing-method
(QualityScore), 8
- FastqQuality-class
(QualityScore-class), 6
- function, 12
- grep, 27, 33, 35, 37, 39

- help, 39
- id (*accessors*), 23
- imageAnalysisPath (*accessors*), 23
- IntegerQuality (*QualityScore*), 8
- IntegerQuality-class
(*QualityScore-class*), 6
- is, 14
- laneDescription (*accessors*), 23
- laneNames (*accessors*), 23
- laneNames, AnnotatedDataFrame-method
(*SolexaSet-class*), 22
- laneNames, SolexaSet-method
(*SolexaSet-class*), 22
- lapply, 15
- lapply, SRList-method
(*SRUtil-class*), 14
- length, MatrixQuality-method
(*QualityScore-class*), 6
- length, QualityScore-method
(*QualityScore-class*), 6
- length, ShortRead-method
(*ShortRead-class*), 16
- length, SRList-method
(*SRUtil-class*), 14
- list.files, 27
- MatrixQuality (*QualityScore*), 8
- MatrixQuality-class
(*QualityScore-class*), 6
- name (*SRFilter-class*), 12
- name, SRFilter-method
(*SRFilter-class*), 12
- names, SRList-method
(*SRUtil-class*), 14
- names<-, SRList, character-method
(*SRUtil-class*), 14
- nFilter (*srFilter*), 41
- NumericQuality, 6, 8
- NumericQuality (*QualityScore*), 8
- NumericQuality-class
(*QualityScore-class*), 6
- OptionalFunction, 12
- pairwiseAlignment, 45
- pileup, 29, 36
- polyn (*Utilites*), 31
- polynFilter (*srFilter*), 41
- position (*accessors*), 23
- PossibleMethod, 12
- qa, 6, 19, 31, 40
- qa, character-method (*qa*), 31
- qa, SolexaPath-method
(*SolexaPath-class*), 20
- QualityScore, 8, 8, 26
- QualityScore-class, 6
- qualPath (*RochePath-class*), 9
- rank, 46
- read454 (*RochePath-class*), 9
- read454, RochePath-method
(*RochePath-class*), 9
- readAligned, 2, 4, 8, 32
- readAligned, character-method
(*readAligned*), 32
- readAligned, SolexaPath-method
(*SolexaPath-class*), 20
- readAligned, SolexaSet-method
(*SolexaSet-class*), 22
- readBaseQuality, 35
- readBaseQuality, character-method
(*readBaseQuality*), 35
- readBaseQuality, SolexaPath-method
(*SolexaPath-class*), 20
- readBfaToc, 29, 36
- readCount (*SRSet-class*), 13
- readData (*SRSet-class*), 13
- readFasta (*RochePath-class*), 9
- readFasta, character-method
(*RochePath-class*), 9
- readFasta, RochePath-method
(*RochePath-class*), 9
- readFastq, 8, 18, 37
- readFastq, character-method
(*readFastq*), 37
- readFastq, SolexaPath-method
(*SolexaPath-class*), 20
- readIndex (*SRSet-class*), 13
- readPath (*RochePath-class*), 9
- readPrb, 36, 38
- readPrb, character-method
(*readPrb*), 38
- readPrb, SolexaPath-method
(*SolexaPath-class*), 20
- readQual (*RochePath-class*), 9
- readQual, character-method
(*RochePath-class*), 9
- readQual, RochePath-method
(*RochePath-class*), 9
- readXStringColumns, 34, 36, 39
- report, 31, 40
- report, character-method (*report*),
40

- report, SolexaExportQA-method
(*SolexaExportQA-class*), 19
- report, SolexaPath-method
(*SolexaPath-class*), 20
- RochePath, 9
- RochePath (*RochePath-class*), 9
- RochePath-class, 9
- RocheSet, 10, 13
- RocheSet (*RocheSet-class*), 11
- RocheSet, character-method
(*RochePath-class*), 9
- RocheSet, RochePath-method
(*RochePath-class*), 9
- RocheSet-class, 11
- runNames (*RochePath-class*), 9
- runNames, RochePath-method
(*RochePath-class*), 9

- sapply, 15
- sapply, SRList-method
(*SRUtil-class*), 14
- scanPath (*accessors*), 23
- SFastqQuality, 21, 38
- SFastqQuality (*QualityScore*), 8
- SFastqQuality, BStringSet-method
(*QualityScore*), 8
- SFastqQuality, character-method
(*QualityScore*), 8
- SFastqQuality, missing-method
(*QualityScore*), 8
- SFastqQuality-class
(*QualityScore-class*), 6
- ShortRead, 3, 18, 46
- ShortRead-class, 16
- ShortReadBase-package, 17
- ShortReadQ, 3, 10, 17, 21, 24, 29, 36, 37
- ShortReadQ-class, 18
- show, 28, 29
- show, .ShortReadBase-method
(*ShortReadBase-package*), 17
- show, AlignedRead-method
(*AlignedRead-class*), 2
- show, ExperimentPath-method
(*ExperimentPath-class*), 5
- show, FastqQuality-method
(*QualityScore-class*), 6
- show, NumericQuality-method
(*QualityScore-class*), 6
- show, RochePath-method
(*RochePath-class*), 9
- show, ShortRead-method
(*ShortRead-class*), 16
- show, SolexaExportQA-method
(*SolexaExportQA-class*), 19
- show, SolexaPath-method
(*SolexaPath-class*), 20
- show, SolexaSet-method
(*SolexaSet-class*), 22
- show, SRFilter-method
(*SRFilter-class*), 12
- show, SRList-method
(*SRUtil-class*), 14
- show, SRSet-method (*SRSet-class*),
13
- show, SRVector-method
(*SRUtil-class*), 14
- SolexaExportQA, 6, 21, 32, 41
- SolexaExportQA
(*SolexaExportQA-class*), 19
- SolexaExportQA-class, 19
- SolexaPath, 5, 22, 23, 32, 33, 38
- SolexaPath (*SolexaPath-class*), 20
- solexaPath (*accessors*), 23
- SolexaPath-class, 20
- SolexaSet, 21, 33
- SolexaSet (*SolexaSet-class*), 22
- SolexaSet, character-method
(*SolexaSet-class*), 22
- SolexaSet, SolexaPath-method
(*SolexaPath-class*), 20
- SolexaSet-class, 22
- sourcePath (*SRSSet-class*), 13
- sprintf, 14
- srapply, 38, 43
- srdistance, 17, 42, 45
- srdistance, DNASTringSet, character-method
(*srdistance*), 45
- srdistance, DNASTringSet, DNASTring-method
(*srdistance*), 45
- srdistance, DNASTringSet, DNASTringSet-method
(*srdistance*), 45
- srdistance, ShortRead, ANY-method
(*ShortRead-class*), 16
- srdistanceFilter (*srFilter*), 41
- sruplicated, 46
- sruplicated, AlignedRead-method
(*AlignedRead-class*), 2
- sruplicated, FastqQuality-method
(*QualityScore-class*), 6
- sruplicated, ShortRead-method
(*ShortRead-class*), 16
- sruplicated, XStringSet-method
(*sruplicated*), 46
- sread, 46

- sread (*accessors*), 23
- SRError (*SRUtil-class*), 14
- SRError-class (*SRUtil-class*), 14
- SRFilter, 21, 23, 33, 41–43
- srFilter, 12, 41
- srFilter, function-method (*srFilter*), 41
- srFilter, missing-method (*srFilter*), 41
- srFilter, SRFilter-method (*SRFilter-class*), 12
- SRFilter-class, 12
- SRList, 19
- SRList (*SRUtil-class*), 14
- SRList-class (*SRUtil-class*), 14
- srorder (*sruplicated*), 46
- srorder, AlignedRead-method (*AlignedRead-class*), 2
- srorder, FastqQuality-method (*QualityScore-class*), 6
- srorder, ShortRead-method (*ShortRead-class*), 16
- srorder, XStringSet-method (*sruplicated*), 46
- srrank (*sruplicated*), 46
- srrank, AlignedRead-method (*AlignedRead-class*), 2
- srrank, FastqQuality-method (*QualityScore-class*), 6
- srrank, ShortRead-method (*ShortRead-class*), 16
- srrank, XStringSet-method (*sruplicated*), 46
- SRSet, 11
- SRSet-class, 13
- rsort, 8
- rsort (*sruplicated*), 46
- rsort, FastqQuality-method (*QualityScore-class*), 6
- rsort, ShortRead-method (*ShortRead-class*), 16
- rsort, XStringSet-method (*sruplicated*), 46
- SRUtil-class, 14
- SRVector (*SRUtil-class*), 14
- SRVector-class (*SRUtil-class*), 14
- SRWarn (*SRUtil-class*), 14
- SRWarn-class (*SRUtil-class*), 14
- strand, AlignedRead-method (*AlignedRead-class*), 2
- strandFilter (*srFilter*), 41
- tables, 17, 47
- tables, ShortRead-method (*ShortRead-class*), 16
- tables, XStringSet-method (*tables*), 47
- Utilites, 31
- vclass (*accessors*), 23
- Versioned, 1
- width, FastqQuality-method (*QualityScore-class*), 6
- width, MatrixQuality-method (*QualityScore-class*), 6
- width, NumericQuality-method (*QualityScore-class*), 6
- width, QualityScore-method (*QualityScore-class*), 6
- width, ShortRead-method (*ShortRead-class*), 16
- XStringSet, 25, 46, 47