ShortRead
November 11, 2009

R topics documented:

accessors .......................................................... 2
AlignedDataFrame-class ........................................... 3
AlignedDataFrame .................................................. 4
AlignedRead-class .................................................. 5
AlignedRead ......................................................... 6
alphabetByCycle ................................................... 7
alphabetScore ....................................................... 9
BowtieQA-class ..................................................... 9
clean ................................................................. 10
countLines .......................................................... 11
deprecated .......................................................... 12
detail ................................................................. 12
dustyScore .......................................................... 13
ExperimentPath-class ................................................ 14
FastqQA-class ....................................................... 15
Intensity-class ...................................................... 16
MAQMapQA-class ................................................... 17
pileup ............................................................... 18
Utilites .............................................................. 19
.QA-class ............................................................ 20
qa .................................................................... 21
QualityScore-class .................................................. 22
QualityScore ......................................................... 24
readAligned .......................................................... 25
readBaseQuality .................................................... 29
readBfaToc .......................................................... 30
readFastq ............................................................ 31
readIntensities ..................................................... 33
readPrb .............................................................. 34
readQseq ............................................................. 35
readXStringColumns ............................................... 36
report ............................................................... 37
RochePath-class .................................................... 38
RocheSet-class ..................................................... 40
ShortRead-class .................................................... 41
ShortReadBase-package ............................................ 42
ShortReadQ-class ................................................... 43
Accessors for ShortRead classes

Description

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

Usage

```r
## 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, ...)
```
**AlignedDataFrame-class**

**Arguments**

object  
An object derived from class ShortRead. See help pages for individual objects, e.g., ShortReadQ. The default is to extract the contents of a slot of the corresponding name (e.g., slot sread) from object.

...  
Additional arguments passed to the accessor. The default definitions do not make use of additional arguments.

**Value**

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

**Author(s)**

Martin Morgan

**Examples**

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

---

**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.
AlignedDataFrame

Extends

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

Methods

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

Additional methods include:

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

Author(s)

Martin Morgan <mtmorgan@fhcrc.org>

See Also

AnnotatedDataFrame

AlignedDataFrame

AlignedDataFrame constructor

Description

Construct an AlignedDataFrame from a data frame and its metadata

Usage

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

Arguments

<table>
<thead>
<tr>
<th>Argument</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>data</td>
<td>A data frame containing alignment information.</td>
</tr>
<tr>
<td>metadata</td>
<td>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.</td>
</tr>
<tr>
<td>nrow</td>
<td>An optional argument, to be used when data is not provided, to construct an AlignedDataFrame with the specified number of rows.</td>
</tr>
</tbody>
</table>

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 "DNAStringSet" DNA sequence of the read.
- `id`: Object of class "BStringSet" read identifier.

Extends

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

Methods

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

- `[ signature(x = "AlignedRead", i = "ANY", j = "missing")`: This method creates a new `AlignedRead` object containing only those reads indexed by i.chromosome is recoded to contain only those levels in the new subset.
- `append signature(x = "AlignedRead", values = "AlignedRead", length = "missing")`: append values after x.chromosome and strand must be factors with the same levels. See methods for ShortReadQ, AlignedDataFrame for details of how these components of x and y are appended.
- `coerce signature(from = "PairwiseAlignedXStringSet", to = "AlignedRead")`: (Invoke this method with, as(from, "AlignedRead")) coerce objects of class from to class to.
- `strand signature(object = "AlignedRead")`: access the strand slot of object.
alignedRead

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.
The return value of coverage is a GenomeData object.
srorder signature(x = "AlignedRead"): 
srrank signature(x = "AlignedRead"): 
srsort signature(x = "AlignedRead"): 
srduplicated 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@fhrcc.org>

See Also
readAligned

Examples

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

AlignedRead Construct objects of class "AlignedRead"

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

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

sread  An object of class DNAStringSet, containing the DNA sequences of the short reads.

id  An object of class BStringSet, containing the identifiers of the short reads. This object is the same length as sread.

quality  An object of class BStringSet, containing the ASCII-encoded quality scores of the short reads. This object is the same length as sread.

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

position  A integer vector describing the (base pair) position at which each short read begins its alignment.

strand  A factor describing the strand to which the short read aligns.

alignQuality  A numeric vector describing the alignment quality.

alignData  An AlignedDataFrame with number of rows equal to the length of sread, containing additional information about alignments.

Value

An object of class AlignedRead.

Author(s)

Martin Morgan <mtmorgan@fhcrc.org>

See Also

AlignedRead.

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, ...)
alphabetByCycle

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.


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

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

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

**Usage**

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

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

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

**Author(s)**
Martin Morgan <mtmorgan@fhcrc.org>

---

**BowtieQA-class**  
*Quality assessment summaries from Bowtie 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, with the argument `type="Bowtie"`.

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

**Extends**
Class "SRList", directly. Class "\.QA\", directly. Class "\.SRUtil\", by class "SRList", distance 2. Class "\.ShortReadBase\", by class "\.QA\", distance 2.
Methods

Accessor methods are inherited from the \texttt{SRList} class.

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

Author(s)

Martin Morgan <mtmorgan@fhcrc.org>

See Also

\texttt{qa}.

Examples

\texttt{showClass("BowtieQA")}

\begin{verbatim}
  clean         Remove sequences with ambiguous nucleotides from short read classes
\end{verbatim}

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

\begin{verbatim}
  clean(object, ...)
\end{verbatim}

Arguments

\begin{verbatim}
  object    An object for which clean methods exist; see below to discover these methods.
  ...       Additional arguments, perhaps used by methods.
\end{verbatim}

Details

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

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

Value

An instance of \texttt{class(object)}, containing only sequences with non-redundant nucleotides.

Author(s)

Martin Morgan <mtmorgan@fhcrc.org>

Examples

\texttt{showMethods('clean')}
countLines  

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

Description

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

Usage

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

Arguments

dirPath  A character vector (or other object; see methods defined on this generic) giving the directory path (relative or absolute) of files whose lines are to be counted.

pattern  The (grep-style) pattern describing files whose lines are to be counted. The default (character(0)) results in line counts for all files in the directory.

...  Additional arguments, passed internally to list.files. See list.files.

useFullName  A logical(1) indicating whether elements of the returned vector should be named with the base (file) name (default; useFullName=FALSE) or the full path name (useFullName=TRUE).

Value

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

Author(s)

Martin Morgan

Examples

sp <- SolexaPath(system.file('extdata', package='ShortRead'))
countLines(analysisPath(sp))
countLines(experimentPath(sp), recursive=TRUE)
countLines(experimentPath(sp), recursive=TRUE, useFullName=TRUE)
**deprecated**  
*Deprecated and defunct functions*

**Description**  
These functions were introduced but are now deprecated or defunct.

**Usage**

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

**Arguments**

- `object`  
  For `basePath`, and `object` of class `ExperimentPath`.  
- `...`  
  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**

```r
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
`dustyScore` Summarize low-complexity sequences

### Description

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

### Usage

```r
dustyScore(x, ...)
```

### Arguments

- **x**: A `DNAStringSet` object, or object derived from `ShortRead`, containing a collection of reads to be summarized.
- **...**: Additional arguments, not currently used.

### Details

The following methods are defined:

- **`dustyScore` signature**: operating on an object derived from class `DNAStringSet`.
- **`dustyScore` signature**: operating on the `sread` of an object derived from class `ShortRead`.

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

### Value

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

### Author(s)

Herve Pages (code); Martin Morgan

### References

See Also

windowmasker/windowmasker_suppl.pdf

Examples

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

---

**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.
**FastqQA-class**

**Author(s)**

Michael Lawrence

**Examples**

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

---

**FastqQA-class**

Quality assessment summaries from MAQ map 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**


**Methods**

- Accessor methods are inherited from the `SRList` class.
- Additional methods defined on this class are:
  
  ```r
  report signature(x="FastqQA", ..., dest=tempfile(), type="html")
  ```

  produces HTML files summarizing QA results. `dest` should be a directory.

**Author(s)**

Martin Morgan <mtmorgan@fhcrc.org>

**See Also**

`qa`

**Examples**

```r
showClass("FastqQA")
```
Intensity-class

"Intensity", "IntensityInfo", and "IntensityMeasure" base classes for short read image intensities

Description

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

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

Objects from the Class

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

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

Slots

Class Intensity has slots:

- **readInfo**: Object of class "IntensityInfo" containing columns for the lane, tile, x, and y coordinates of the read.
- **intensity**: Object of class "IntensityMeasure" containing image intensity data for each read and cycle.
- **measurementError**: Object of class "IntensityMeasure" containing measures of image intensity uncertainty for each read and cycle.
- **hasMeasurementError**: Length 1 logical variable indicating whether intensity standard errors are included (internal use only).

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

Extends

These classes extend ".ShortReadBase", directly.

Methods

Methods and accessor functions for Intensity include:

- **readInfo** signature(object = "Intensity"): access the readInfo slot of object.
- **intensity** signature(object = "Intensity"): access the intensity slot of object.
- **measurementError** signature(object = "Intensity"): access the nse slot of object, or signal an error if no standard errors are available.
- **dim** signature(object = "Intensity"): return the dimensions (e.g., number of reads by number of cycles) represented by object.
show signature(object = "Intensity"): provide a compact representation of the object.

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

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

Author(s)

Martin Morgan <mtmorgan@fhcrc.org>

See Also

`readIntensities`

Examples

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

**MAQMapQA-class**  
Quality assessment summaries from MAQ map 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**


**Methods**

Accessor methods are inherited from the `SRList` class.

`report` signature(x="MAQMapQA", ..., dest=tempfile(), type="html"): produces an html file summarizing the QA results.
Author(s)
Martin Morgan <mtmorgan@fhcrc.org>

See Also
qa.

Examples
showClass("MAQMapQA")

---

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 = strand("+"),
       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 an 'AlignedRead' object containing aligned reads froma 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 repored by 'width(lane)') is only 24, 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 by:

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

## End(Not run)

Utilities

Utilities for common, simple operations

Description

These functions perform a variety of simple operations.

Usage

```
polyn(nucleotides, n)
```
Arguments

nucleotides A character vector with all elements having exactly 1 character, typically from the IUPAC alphabet.
n An integer(1) vector.

Details

polyn returns a character vector with each element having n characters. Each element contains a single nucleotide. Thus polyn("A", 5) returns AAAAA.

Value

polyn returns a character vector of length length(nucleotide)

Author(s)

Martin Morgan <mtmorgan@fhcrc.org>

Examples

polyn(c("A", "N"), 35)

.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

gClass(".QA", where=getNamespace("ShortRead"))
Perform quality assessment on short reads

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

Usage
qa(dirPath, ...)

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:

- `dirPath = "character", pattern = "character", type = c("SolexaExport", "Bowtie", "MAQMapShort", "fastq"), ...`  Quality assessment is performed on all files in directory `dirPath` whose file name matches `pattern`. The type of analysis performed is based on the `type` argument. Use `SolexaExport` when all files matching `pattern` are Solexa_export.txt files. Use `Bowtie` for Bowtie files. Use `MAQMapShort` for MAQ map files produced by MAQ versions below 0.70, and `fastq` for collections of fastq-format files. Quality assessment details vary depending on data source.

Value
An object derived from class `.QA`

Author(s)
Martin Morgan <mtmorgan@fhcrc.org>

See Also
`.QA, SolexaExport, QA MAQMap, QA FastqQA`

Examples
showMethods("qa")
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.

dim signature(x = "MatrixQuality"):
    The integer(2) dimension (e.g., number of reads, read width) represented by the quality score.

length signature(x = "QualityScore"):

length signature(x = "MatrixQuality"):
    The integer(1) length (e.g., number of reads) represented by the quality score. Note that length of MatrixQuality is the number of rows of the corresponding matrix, and not the length of the corresponding numeric vector.

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

width signature(x = "QualityScore"):

width signature(x = "NumericQuality"):

width signature(x = "MatrixQuality"):

width signature(x = "FastqQuality"):
    A numeric vector with length equal to the number of quality scores, and value equal to the number of quality scores for each read. For instance, a FastqQuality will have widths equal to the number of nucleotides in the underlying short read.
show signature(object = "QualityScore"):  
show signature(object = "NumericQuality"):  
show signature(object = "FastqQuality"):  
  provide a brief summary of the object content.
detail signature(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.
coerce signature(from = "FastqQuality", to = "numeric"):  
coerce signature(from = "FastqQuality", to = "matrix"):  
coerce signature(from = "SFastqQuality", to = "matrix"):  
  (Use these as, for instance, as(from, "matrix") to coerce objects of class from to class 
  to, using the quality encoding implied by the class. When to is "matrix", the result is a 
  matrix of type integer. In addition, methods require that all quality scores are of the same 
  width.
narrow signature(x = "FastqQuality", start = NA, end = NA, width = NA, 
  use.names = TRUE): 'narrow' quality so that scores are between start and end 
  bases, according to narrow in the IRanges package.
srorder signature(x = "FastqQuality"):  
srrank signature(x = "FastqQuality"):  
srduplicated signature(x = "FastqQuality"):  
  Apply srsort, srorder, srrank, and srduplicated to quality scores, returning ob-
  jects as described on the appropriate help page.

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

Author(s)

Martin Morgan <mtmorgan@fhcrc.org>

See Also

NumericQuality and other constructors.

Examples

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

Construct objects indicating read or alignment quality

Description

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

Usage

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

Arguments

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

... Additional arguments, currently unused.

Value

Constructors return objects of the corresponding class derived from QualityScore.

Author(s)

Martin Morgan <mtmorgan@fhcrc.org>

See Also

QualityScore, readFastq, readAligned

Examples

nq <- NumericQuality(rnorm(20))
nq
quality(nq)
quality(nq[10:1])
readAligned is a function in R that reads aligned reads and their quality scores into R representations. It reads all aligned read files in a directory specified by `dirPath` whose file name matches the pattern specified by `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**

```r
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 (character(0)) results in (attempted) input of all files in the directory.
- `...`: Additional arguments, used by methods. When `dirPath` is a character vector, the argument `type` must be provided. Possible values for `type` and their meaning are described below. Most methods implement `filter=srFilter()`, allowing objects of `SRFilter` to selectively return 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`
readAligned

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 SolexaAlign
**type**="SolexaRealign"  These types parse s_L_TTTT_prealign.txt, s_L_TTTT_align.txt, or s_L_TTTT_realign.txt files produced by default and eland analyses. From the Solexa documentation, align corresponds to unfiltered first-pass alignments, prealign adjusts alignments for error rates (when available), realign filters alignments to exclude clusters failing to pass quality criteria. Because base quality scores are not stored with alignments, the object returned by readAligned scores all base qualities as -32.
If parsed, values can be retrieved from AlignedRead as follows:

Sequence  stored in sread
Best score  stored in alignQuality
Number of hits  stored in alignData
Target position  stored in position
Strand  stored in strand
Target sequence  Ignored; parse using readXStringColumns
Next best score  stored in alignData

**type**="SolexaResult"  This parses s_L_eland_results.txt files, an intermediate format that does not contain read or alignment quality scores. Because base quality scores are not stored with alignments, the object returned by readAligned scores all base qualities as -32.
Columns of this file type can be retrieved from AlignedRead as follows (description of columns is from Table 19, Genome Analyzer Pipeline Software User Guide, Revision A, January 2008):

Id  Not parsed
Sequence  stored in sread

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

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

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

Genome file of match stored in `chromosome`

Position stored in `position`

Strand (direction of match) stored in `strand`

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

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

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

```
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 `mapiew` command. Interpretation of columns is based on the description in the MAQ manual, specifically

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

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

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

Remaining fields are currently ignored.

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

```
Identifier id
Strand strand
Chromosome chromosome
Position position; see comment below
Read sread; see comment below
Read quality quality; see comments below
Bowtie reserved ignored
Alignment mismatch locations alignData
```
This method includes the argument `qualityType` to specify how quality scores are encoded. Bowtie quality scores are ‘Solexa’-like by default, with `qualityType='SFastqQuality'`, but can be specified as ‘Phred’-like, with `qualityType='FastqQuality'`.

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

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

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

- `id` id
- `seq` sread; see comment below
- `qual` quality; see comment below
- `number of hits` alignData
- `a/b` alignData (pairedEnd)
- `length` alignData (alignedLength)
- `+/−` strand
- `chr` chromosome
- `location` position; see comment below
- `types` alignData (typeOfHit: integer portion; hitDetail: text portion)

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

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

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

Value

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

Author(s)

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

See Also

A `AlignedRead` object.


readBaseQuality

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

dirPath A character vector (or other object; see methods defined on this generic) giving
the directory path (relative or absolute) of files to be input.

seqPattern The (grep-style) pattern describing base call file names to be read. The default
(^(character(0)) results in (attempted) input of all files in the directory.

prbPattern The (grep-style) pattern describing quality score file names to be read. The de-
default (^(character(0)) results in (attempted) input of all files in the directory.

... 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.
readBfaToc

**Author(s)**

Patrick Aboyoun <paboyoun@fhcrc.org>

**See Also**

A ShortReadQ object.

readXStringColumns, readPrb

**Examples**

```r
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**

```r
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

**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.

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

**Usage**

```r
readFastq(dirPath, pattern=character(0), ...)
## S4 method for signature 'character':
readFastq(dirPath, pattern=character(0), ..., withIds=TRUE)
writeFastq(object, file, mode="w", ...)
```

**Arguments**

- `dirPath`: 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.
- `pattern`: The (grep-style) pattern describing file names to be read. The default (character(0)) results in line (attempted) input of all files in the directory.
- `object`: An object to be output in fastq format. For methods, use `showMethods(object, where=getNamespace("ShortRead"))`.
- `file`: A length 1 character vector providing a path to a file to the object is to be written to.
- `mode`: A length 1 character vector equal to either ‘w’ or ‘a’ to write to a new file or append to an existing file, respectively.
- `...`: Additional arguments, perhaps used by methods.
- `withIds`: logical(1) indicating whether identifiers should be read from the fastq file.

**Details**

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

```
@HWI-EAS88_1_1_1_1001_499
GGACCTTTGTAGGATACCCCTGCTTCTTCTCCTGT
+HWI-EAS88_1_1_1_1001_499
[[[[[[[[[[[[[[[yy]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]VCHVMPAS
```

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

The encoding implicit in Solexa-derived fastq files is that each character code corresponds to a score equal to the ASCII character value minus 64 (e.g., ASCII @ is decimal 64, and corresponds to a Solexa quality score of 0). This is different from BioPerl, for instance, which recovers quality scores by subtracting 33 from the ASCII character value (so that, for instance, !, with decimal value 33, encodes value 0).

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

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

Value

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

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

Author(s)

Martin Morgan

See Also

The IUPAC alphabet in Biostrings.


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

file <- tempfile()
writeFastq(rfq1, file)
readLines(file, 8)
readIntensities  

Description

readIntensities reads image ‘intensity’ files (such as Solexa’s _int.txt and (optionally) _nse.txt) in a directory into a single object.

Usage

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

Arguments

dirPath  Directory path or other object (e.g., SolexaPath for which methods are defined.
pattern  A length 1 character vector representing a regular expression to be combined (using, e.g., paste(pattern, intExtension, sep=" ") with intExtension or nseExtension to match files to be summarized.
...  Additional arguments used by methods.

Details

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

The readIntensities,character-method contains an arugment type that determines how intensities are parsed. Use the type argument to readIntensities,character-method, as follows:

**IparIntensity**  Intensities are read from Solexa_pos.txt, _int.txt.p, _nse.txt.p-style file triplets.  The signature for this method is
dirPath, pattern=character(0), ..., type="IparIntensity", intExtension="_int.txt.p", nseExtension="_nse.txt.p.gz", posExtension="_pos.txt", withVariability=TRUE, verbose=FALSE

**SolexaIntensity**  Intensities are read from Solexa _int.txt and _nse.txt-style files; see SolexaPath for details. The signature for this method is
dirPath, pattern=character(0), ..., type="SolexaIntensity", intExtension="_int.txt", nseExtension="_nse.txt", withVariability=TRUE, verbose=FALSE

Arguments to these methods are as follows:

**intExtension, nseExtension, posExtension**  character(1) values pasted (with sep=" ") to pattern to identify different file sources.

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

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

Value

An object derived from class Intensity.

Author(s)

Martin Morgan <mtmorgan@fhcrc.org>
Examples

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

readPrb

**Read Solexa prb files as fastq-style quality scores**

Description

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

Usage

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

Arguments

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

Details

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

- `as="SolexaEncoding"` The ASCII encoding of the maximum per cycle and read quality score is encoded using Solexa conventions.
- `as="FastqEncoding"` The ASCII encoding of the maximum per cycle and read quality score is encoded using Fastq conventions, i.e., ! has value 0.
- `as="IntegerEncoding"` The maximum per cycle and read quality score is returned as a in integer value. Values are collated into a matrix with number of rows equal to number of reads, and number of columns equal to number of cycles.
- `as="array"` The quality scores are not summarized; the return value is an integer array with dimensions corresponding to reads, nucleotids, and cycles.

Value

An object of class `QualityScore`, or an integer matrix.
readQseq

Author(s)

Martin Morgan <mtmorgan@fhcrc.org>

Examples

fl <- system.file("extdata", package="ShortRead")
sp <- SolexaPath(fl)
readPrb(sp, "s_1_*_prb.txt") # all tiles to a single file

Description

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

Usage

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

Arguments

dirPath Directory path or other object (e.g., SolexaPath) for which methods are defined.

pattern Regular expression matching names of _qseq files to be summarized.

... Additional argument, passed to I/O functions.

as character(1) indicating the class of the return type.

filtered logical(1) indicating whether to include only those reads passing Solexa filtering?

verbose logical(1) indicating whether to report on progress during evaluation.

Value

An object of class ShortReadQ.

Author(s)

Martin Morgan <mtmorgan@fhcrc.org>

Examples

fl <- system.file("extdata", package="ShortRead")
sp <- SolexaPath(fl)
readQseq(sp)
**readXStringColumns**  
*Read one or more columns into XStringSet (e.g., DNAStringSet) objects*

**Description**

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

**Usage**

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

**Arguments**

- **dirPath**
  A character vector giving the directory path (relative or absolute) of files to be read.

- **pattern**
  The (grep-style) pattern describing file names to be read. The default (character(0)) reads all files in dirPath. All files are expected to have identical numbers of columns.

- **colClasses**
  A list of length equal to the number of columns in a file. Columns with corresponding colClasses equal to NULL are ignored. Other entries in colClasses are expected to be character strings describing the base class for the XStringSet. For instance a column of DNA sequences would be specified as "DNAString". The column would be parsed into a DNAStringSet object.

- **nrows**
  A length 1 integer vector describing the maximum number of XString objects to read into the set. Reads may come from more than one file when dirPath and pattern parse several files and nrows is greater than the number of reads in the first file.

- **skip**
  A length 1 integer vector describing how many lines to skip at the start of each file.

- **sep**
  A length 1 character vector describing the column separator.

- **header**
  A length 1 logical vector indicating whether files include a header line identifying columns. If present, the header of the first file is used to name the returned values.

- **comment.char**
  A length 1 character vector, with a single character that, when appearing at the start of a line, indicates that the entire line should be ignored. Currently there is no way to use comment characters in other than the first position of a line.

**Value**

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

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

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

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

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

### Description

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

### Usage

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

### Arguments

- **x**: An object returned by `qa`, usually derived from class `.QA`
- **...**: Additional arguments used by specific methods. See specific methods for details.
- **dest**: The output destination for the final report. For `type="html"` this is a directory; for (deprecated) `type="pdf"` this is a file.
- **type**: A text string defining the type of report; available report types depend on the type of object `x`; usually this is “html”.

### Details

The following methods are defined:

- `tempfile(), type="html"` Produce an HTML-based report from an object of class `BowtieQA`.
- `tempfile(), type="html"` Produce an HTML-based report from an object of class `FastqQA`.
- `tempfile(), type="html"` Produce an HTML-based report from an object of class `MAQMapQA`.
- `tempfile(), type="html"` Produce an HTML-based report from an object of class `SolexaExportQA`.
- `tempfile(), type="pdf"` (Deprecated) Produce an PDF report from an object of class `SolexaExportQA`.
Produce an HTML report by first visiting all _export.txt files in the analysisPath directory of x to create a SolexaExportQA instance.

(Deprecated) Produce an PDF report by first visiting all _export.txt files in the analysisPath directory of x to create a SolexaExportQA instance.

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")

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 RocehPath 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.
RochePath-class

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 DNAStringSet.
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**

**Roche (454) experiment-wide data container**

**Description**

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

**Objects from the Class**

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

**Slots**

- **sourcePath**: Object of class "RochePath" The file system location of the data used in this experiment.
- **readIndex**: Object of class "integer" indexing reads included in the experiment; see `SRSet` for details on data representation in this class.
- **readCount**: Object of class "integer" containing the number of reads associated with each sample; see `SRSet` for details on data representation in this class.
- **phenoData**: Object of class "AnnotatedDataFrame" with as many rows as there are samples, containing information on experimental design.
- **readData**: Object of class "AnnotatedDataFrame" containing as many rows as there are reads, containing information on each read in the experiment.

**Extends**

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

**Methods**

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

**Author(s)**

Michael Lawrence <mflawrence@fhcrc.org>

**See Also**

`SRSet`

**Examples**

showClass("RocheSet")
ShortRead-class

"ShortRead" class for short reads

Description

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

Objects from the Class

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

Slots

- **sread**: Object of class "DNAStringSet" containing IUPAC-standard, uniform-length DNA strings represent short sequence reads.
- **id**: Object of class "BStringSet" containing identifiers, one for each short read.

Extends

Class ".ShortReadBase", directly.

Methods

Constructors include:

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

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.
- **append signature** (x = "ShortRead", values = "ShortRead", length = "missing"): append the sread and id slots of values after the corresponding fields of x.
- **narrow signature** (x = "ShortRead", start = NA, end = NA, width = NA, use.names = TRUE): ‘narrow’ sread so that sequences are between start and end bases, according to narrow in the IRanges package.
- **length signature** (x = "ShortRead"): returns a integer(1) vector describing the number of reads in this object.
- **width signature** (x = "ShortRead"): returns an integer() vector of the widths of each read in this object.
srorder signature(x = "ShortRead"):
srrank signature(x = "ShortRead"):
srsort signature(x = "ShortRead"):
srduplicated signature(x = "ShortRead"): Order, rank, sort, and find duplicates in ShortRead objects based on sread(x), analogous to the corresponding functions order, rank, sort, and duplicated, ordering nucleotides in the order ACGT.
srdistance signature(pattern="ShortRead", subject="ANY"): Find the edit distance between each read in pattern and the (short) sequences in subject. See srdistance for allowable values for subject, and for additional details.
trimLRPatterns signature(Lpattern = "", Rpattern = "", subject = "ShortRead", max.Lmismatch = 0, max.Rmismatch = 0, with.Lindels = FALSE, with.Rindels = FALSE, Lfixed = TRUE, Rfixed = TRUE, ranges = FALSE): Remove left and/or right flanking patterns from sread(subject), as described in trimLRPatterns. Classes derived from ShortRead (e.g., ShortReadQ, AlignedRead) have corresponding base quality scores trimmed, too. A user-supplied argument ranges is ignored by this method; the class of the return object is the same as the class of subject.
alphabetByCycle signature(stringSet = "ShortRead"): Apply alphabetByCycle to the sread component of stringSet, returning a matrix as described in alphabetByCycle.
tables signature(x= "ShortRead", n = 50): Apply tables to the sread component of x, returning a list summarizing frequency of reads in x.
clean signature(object="ShortRead"): Remove all reads containing non-nucleotide ("N", ".") symbols.
show signature(object = "ShortRead"): provides a brief summary of the object, including its class, length and width.
detail signature(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")

Base classes and methods for 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 the result of readFastq, or can be constructed from DNAStringSet, QualityScore, and BStringSet objects, as described below.

Slots

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

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

Extends

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

Methods

Constructors include:

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

ShortReadQ signature(sread = "DNAStringSet", quality = "QualityScore", id = "missing"): Create a ShortReadQ object from reads and their quality scores, creating empty identifiers.

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

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

writeFastq signature(object = "ShortReadQ", file = "character", mode="character", ...): Write object to file in fastq format. mode defaults to 'w'. This creates a new file, or fails if file already exists. Use mode="a" to append to an existing file. file is expanded using path.expand.
[ 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.

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

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

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)
**SolexaIntensity-class**

**Extends**


**Methods**

Accessor methods are inherited from the SRList class.

Additional methods defined on this class are:

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

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

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

**Author(s)**

Martin Morgan <mtmorgan@fhcrc.org>

**See Also**

qa.

**Examples**

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

---

**SolexaIntensity-class**

*Classes "SolexaIntensity" and "SolexaIntensityInfo"*

**Description**

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

**Objects from the Class**

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

Slots

Object of SolexaIntensity have slots:

- **readInfo**: Object of class "SolexaIntensityInfo" representing information about each read.
- **intensity**: Object of class "ArrayIntensity" containing an array of intensities with dimensions read, base, and cycle. Nucleotide are A, C, G, T for each cycle.
- **measurementError**: Object of class "ArrayIntensity" containing measurement errors for each read, cycle, and base, with dimensions like that for *intensity*.
- **.hasMeasurementError**: Object of class "ScalarLogical" used internally to indicate whether measurement error information is included.

Object of SolexaIntensityInfo

- **data**: Object of class "data.frame", inherited from AnnotatedDataFrame.
- **varMetadata**: Object of class "data.frame", inherited from AnnotatedDataFrame.
- **dimLabels**: Object of class "character", inherited from AnnotatedDataFrame.
- **.__classVersion__**: Object of class "Versions", inherited from AnnotatedDataFrame.
- **.init**: Object of class "ScalarLogical", used internally to indicate whether the user initialized this object.

Extends

Class SolexaIntensity:
- Class "Intensity", directly. Class ".ShortReadBase", by class "Intensity", distance 2.

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

Methods

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

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

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

Author(s)

Martin Morgan <mtmorgan@fhcrc.org>

See Also

- readIntensities
Examples

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

SolexaIntensity  
Construct objects of class "SolexaIntensity" and "SolexaIntensityInfo"

Description

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

Usage

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

Arguments

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

Value

An object of class SolexaIntensity, or SolexaIntensityInfo.
SolexaPath-class

Author(s)

Martin Morgan <mtmorgan@fhcrc.org>

See Also

SolexaIntensity.

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|IPAR)"), baseCallPath=.solexaPath(imageAnalysisPath, "^Bustard"),
analysisPath=.solexaPath(baseCallPath, "^GERALD"), ..., verbose=FALSE)

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

dataPath (optional) Solexa ‘Data’ folder.

scanPath (optional) Solexa GoldCrest image scan path.

imageAnalysisPath (optional) Firecrest image analysis path.

baseCallPath (optional) Bustard base call path.

analysisPath (optional) Gerald analysis pipeline path.

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

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

All paths must be fully-specified.

Slots

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

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:

**readIntensities** signature(dirPath = "SolexaPath", pattern=character(0),
run, ..., intExtension = "_int.txt", nseExtension = ".nse.txt",
withVariability = TRUE, verbose = FALSE):
Use imageAnalysisPath(sp)[run] as the directory path(s) and pattern=character(0)
as the pattern for discovering Soelxa intExtension and nseExtension files, returning
a **SolexaIntensity** object containing intensities, (optionally) standard errors, and read
(lane, tile, x, y coordinates of cluster) information.

**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 contain-
ing the maximum qualities found for each base of each cycle.
The ... argument may include the named argument as. This influences the return value, as
explained on the **readPrb**, character-method page.

**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.

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

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

**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

```r
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

"SolexaSet" coordinating Solexa output locations with sample annotations

Description

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

Objects from the Class

Objects can be created from the constructor:

SolexaSet(path, ...).

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

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

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.txt" as the pattern for discovering Eland-aligned reads in the Solexa 'export' file format. Note that the default method reads all aligned read files into a single object; often one will want to specify a pattern for each lane. Use an object of SRFilter to select specific chromosomes, strands, etc.

Author(s)

Martin Morgan

Examples

showClass("SolexaSet")
showMethods(class="SolexaSet")
## 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 5", "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)
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

```r
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`.

- **FUN**: A function to be applied to each element of `X`. The function must have `...` or named argument `verbose` 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).

- **...**: Additional arguments, passed to `FUN`.

- **fapply**: An optional argument defining an `lapply`-like function to be used in partitioning `X`. See details, below.

- **reduce**: Optional function accepting a list (the result of `fapply` 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.

- **verbose**: Report whether evaluation is local or mpi-based; also forwarded to `FUN`, 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.
srdistance

Author(s)

Martin Morgan <mtmorgan@fhcrc.org>

Examples

```r
## ... 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

Edit distances between reads and a small number of short references

Description

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

Usage

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

Arguments

- `pattern`: An object of class `DNAStringSet` containing reads whose edit distance is desired.
- `subject`: A short character vector, `DNAString` or (small) `DNAStringSet` to serve as reference.
- `...`: Additional arguments, forward to `srapply`.

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.
Order, sort, and find duplicates in XStringSet objects

Description

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

Usage

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

Arguments

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

Details

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

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

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

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

srduplicated signature(x = "XStringSet"): 
  Apply srorder, srrank, srsort, srduplicated to XStringSet objects such as those returned by sread.

srsort signature(x = "ShortRead"): 

srorder signature(x = "ShortRead"): 

srduplicated signature(x = "ShortRead"): 
  Apply srorder, srrank, srsort, srduplicated to XStringSet objects to the sread component of ShortRead and derived objects.

Value

The functions return the following values:

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

Author(s)

Martin Morgan <mtmorgan@fhcrc.org>

Examples

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

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

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

---

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


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

```r
## see ?srFilter
```

---

**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.
srFilter

Usage

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

compose(filt, ..., .name)

idFilter(regex=character(0), fixed=FALSE, exclude=FALSE,
  .name="idFilter")
chromosomeFilter(regex=character(0), fixed=FALSE, exclude=FALSE,
  .name="ChromosomeFilter")
positionFilter(min=-Inf, max=Inf, .name="PositionFilter")
strandFilter(strandLevels=character(0), .name="StrandFilter")
uniqueFilter(withSread=TRUE, .name="UniqueFilter")
nFilter(threshold=0L, .name="CleanNFilter")
polynFilter(threshold=0L, nuc=c("A", "C", "T", "G", "other"),
  .name="PolyNFilter")
dustyFilter(threshold=Inf, .name="DustyFilter")
srdistanceFilter(subject=character(0), threshold=0L,
  .name="SRDistanceFilter")
alignQualityFilter(expr=expression(), .name="AlignQualityFilter")
alignDataFilter(expr=expression(), .name="AlignDataFilter")

Arguments

fun  An object of class function to be used as a filter. fun must accept a single named argument x, and is expected to return a logical vector such that x[fun(x)] selects only those elements of x satisfying the conditions of fun

name  A character(1) object to be used as the name of the filter. The name is useful for debugging and reference.

d filt  A SRFilter object, to be used with additional arguments to create a composite filter.

.name  An optional character(1) object used to over-ride the name applied to default filters.

regex  Either character(0) or a character(1) regular expression used as grep(regex, chromosome(x)) to filter based on chromosome. The default (character(0)) performs no filtering

fixed  logical(1) passed to grep, influencing how pattern matching occurs.

exclude  logical(1) which, when TRUE, uses regex to exclude, rather than include, reads.

min max  numeric(1) value defining the closed interval in which position must be found, min <= position <= max

strandLevels  Either character(0) or character(1) containing strand levels to be selected. ShortRead objects have standard strand levels NA, "+", "−", "*", with NA meaning strand information not available and "*" meaning strand information not relevant.
withSread A logical(1) indicating whether uniqueness includes the read sequence
(withSread=TRUE) or is based only on chromosome, position, and strand
(withSread=FALSE).

threshold A numeric(1) value representing a minimum (srdistanceFilter, alignQualityFilter) or maximum (nFilter, polynFilter, dustyFilter) criterion for the filter. The minima and maxima are closed-interval (i.e., x >= threshold, x <= threshold for some property x of the object being filtered).

nuc A character vector containing IUPAC symbols for nucleotides or the value "other" corresponding to all non-nucleotide symbols, e.g., N.

subject A character() of any length, to be used as the corresponding argument to srdistance.

expr A expression to be evaluated with pData(alignData(x)).

... Additional arguments for subsequent methods; these arguments are not currently used.

Details

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

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

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

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

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

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

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

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

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

nFilter selects elements with fewer than threshold ’N’ symbols in each element of sread(x).

dustyFilter selects elements with high sequence complexity, as characterized by their dustyScore. This emulates the dust command from WindowMaker software.

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)]

SRSer-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 SRSer.
Objects from the Class

Objects of this class are not usually created directly, but instead are created by a derived class, e.g., RocheSet.

Slots

- **sourcePath**: Object of class "ExperimentPath", containing the directory path where sequence files can be found.
- **readIndex**: Object of class "integer" indicating specific sequences included in the experiment.
- **readCount**: Object of class "integer" containing the number of reads in each sample included in the experiment. The sum of this vector is the total number of reads.
- **phenoData**: Object of class "AnnotatedDataFrame" describing each sample in the experiment. The number of rows of phenoData equals the number of elements in readCount.
- **readData**: Object of class "AnnotatedDataFrame" containing annotations on all reads.

Extends

Class ".ShortReadBase", directly.

Methods

- **experimentPath** signature(object = "SRSet") : return the ExperimentPath associated with this object.
- **phenoData** signature(object = "SRSet") : return the phenoData associated with this object.
- **readCount** signature(object="SRSet") :
- **readIndex** signature(object="SRSet") :
- **readData** signature(object="SRSet") :
- **sourcePath** signature(object="SRSet") : Retrieve the corresponding slot from object.
- **show** signature(object = "SRSet") : display the contents of this object.
- **detail** signature(object = "SRSet") : provide more extensive information on the object.

Author(s)

Michael Lawrence <mflawrence@fhcrc.org>

Examples

showClass("SRSet")
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.

```r
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`.

```r
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.

```r
SRVector(..., vclass)
```

- `...` elements all satisfying an `is` relationship with `vclass`, to be placed in SRVector.
- `vclass` the class to which all elements in `...` belong. If `vclass` is missing and `length(list(...))` is greater than zero, then `vclass` is taken to be the class of the first argument of `...`.

SRVector errors:

- `SRVectorClassDisagreement` this error occurs when not all arguments `...` satisfy an ‘is’ relationship with `vclass`. 

Slots

SRError and SRWarn have the following slots defined:

.type: Object of class "character" containing the type of error or warning. .type must come from a pre-defined list of types, see, e.g., ShortRead:::.SRError_types.

.message: Object of class "character" containing a detailed message describing the error or warning.

SRList has the following slot defined:

.srlist: Object of class "list" containing the elements in the list.

SRVector extends SRList, with the following additional slot:

vclass: Object of class "character" naming the type of object all elements of SRVector must be.

Methods

Accessors are available for all slots, and have the same name as the slot, e.g., vclass to access the vclass slot of SRVector. Internal slots (those starting with '.') also have accessors, but these are not exported e.g., ShortRead:::.type.

SRList has the following methods:

length signature(x = "SRList"): return the (integer(1)) length of the SRList.

names signature(x = "SRList"): return a character vector of list element names. The length of the returned vector is the same as the length of x.

names<- signature(x = "SRList", value = "character"): assign value as names for members of x.

[ signature(x = "SRList", i = "ANY", j = "missing"): subset the list using standard R list subset paradigms.

[[ signature(x = "SRList", i = "ANY", j = "missing"): select element 'i' from the list, using standard R list selection paradigms.

lapply signature(X = "SRList", FUN="ANY"): apply a function to all elements of X, with additional arguments interpreted as with lapply.

sapply signature(X = "SRList"): apply a function to all elements of X, simplifying the result if possible. Additional arguments interpreted as with sapply.

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

class(".SRUtil", where=getNamespace("ShortRead"))
ShortRead:::.SRError_types
ShortRead:::.SRWarn_types
detail(SRList(1:5, letters[1:5]))

tryCatch(SRVector(1:5, letters[1:5]),
  SRVectorClassDisagreement=function(err) {
    cat("caught: ", conditionMessage(err), ", \n")
  })

---

tables

Summarize XStringSet read frequencies

Description

This generic summarizes the number of times each sequence occurs in an XStringSet instance.

Usage

tables(x, n=50, ...)

Arguments

x An object for which a tables method is defined.

n An integer(1) value determining how many named sequences will be present in the top portion of the return value.

... Additional arguments available to methods

Details

Methods of this generic summarize the frequency with which each read occurs. There are two components to the summary. The reads are reported from most common to least common; typically a method parameter controls how many reads to report. Methods also return a pair of vectors describing how many reads were represented 1, 2, ... times.

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

tables signature(x = "XStringSet", n = 50): Apply tables to the XStringSet x.

Value

A list of length two.

top A named integer vector. Names correspond to sequences. Values are the number of times the corresponding sequence occurs in the XStringSet. The vector is sorted in decreasing order; methods typically include a parameter specifying the number of sequences to return.

distribution a data.frame with two columns. nOccurrences is the number of times any particular sequence is represented in the set (1, 2, ...). nReads is the number of reads with the corresponding occurrence.
Author(s)

Martin Morgan <mtmorgan@fhcrc.org>

Examples

showMethods("tables")
sp <- SolexaPath(system.file("extdata", package="ShortRead"))
aln <- readAligned(sp)
tables(sread(aln), n=6)
xyplot(log10(nReads)-log10(nOccurrences),
       tables(sread(aln))$distribution)
Index

*Topic IO
  readXStringColumns, 35

*Topic classes
  .QA-class, 19
  AlignedDataFrame-class, 2
  AlignedRead-class, 4
  BowtieQA-class, 8
  ExperimentPath-class, 13
  FastqQA-class, 14
  Intensity-class, 15
  MAQMapQA-class, 16
  QualityScore-class, 21
  RochePath-class, 37
  RocheSet-class, 39
  ShortRead-class, 40
  ShortReadQ-class, 42
  SolexaExportQA-class, 43
  SolexaIntensity-class, 44
  SolexaPath-class, 47
  SolexaSet-class, 49
  SRFilter-class, 54
  SRSet-class, 58
  SRUtil-class, 60

*Topic manip
  accessors, 1
  AlignedDataFrame, 3
  AlignedRead, 5
  alphabetByCycle, 6
  alphabetScore, 8
  clean, 9
  countLines, 10
  deprecated, 11
  detail, 11
  dustyScore, 12
  qa, 20
  QualityScore, 23
  readAligned, 24
  readBaseQuality, 28
  readFastq, 30
  readIntensities, 32
  readPrb, 33
  readQseq, 34
  report, 36
  SolexaIntensity, 46
  srapply, 51
  srdistance, 52
  srduplicate, 53
  srFilter, 55
  tables, 62
  Utilities, 18

*Topic package
  ShortReadBase-package, 41
    .QA, 8, 14, 16, 20, 36, 44
    .QA-class, 19
    .Roche, 38, 39
    .Roche-class
      (ShortReadBase-package), 41
      .SRUtil, 8, 14, 16, 44, 55
      .SRUtil-class (SRUtil-class), 60
      .ShortReadBase, 4, 8, 13–16, 19, 21, 38–40, 42, 44, 45, 48, 50, 59
      .ShortReadBase-class
        (ShortReadBase-package), 41
        .Solexa, 48, 50
        .Solexa-class
          (ShortReadBase-package), 41
          [,AlignedRead, ANY, ANY, ANY-method (AlignedRead-class), 4
          [,AlignedRead, ANY, ANY-method (AlignedRead-class), 4
          [,AlignedRead, ANY, missing, ANY-method (AlignedRead-class), 4
          [,AlignedRead, ANY, missing-method (AlignedRead-class), 4
          [,AlignedRead, missing, ANY, ANY-method (AlignedRead-class), 4
          [,AlignedRead, missing, ANY-method (AlignedRead-class), 4
          [,AlignedRead, missing, missing, ANY-method (AlignedRead-class), 4
          [,AlignedRead, missing, missing-method (AlignedRead-class), 4
          [,IntensityMeasure, ANY, ANY, ANY-method (Intensity-class), 15
          [,IntensityMeasure, ANY, ANY-method (Intensity-class), 15

65
INDEX

[,IntensityMeasure, ANY, missing, ANY-method (Intensity-class), 15
[,IntensityMeasure, missing, ANY, ANY-method (Intensity-class), 15
[,IntensityMeasure, missing, missing, ANY-method (Intensity-class), 15
[,MatrixQuality, ANY, missing, ANY-method, SolexaIntensity, missing, missing, ANY-method (SolexaIntensity-class), 44
[,MatrixQuality, ANY, missing-method (QualityScore-class), 21
[,QualityScore, ANY, missing-method (QualityScore-class), 21
[,QualityScore, ANY, missing-method (QualityScore-class), 21
[,SRList, ANY, missing-method (SRUtil-class), 60
[,SRList, ANY, missing-method (SRUtil-class), 60
[,ShortRead, ANY, ANY, ANY-method (ShortRead-class), 40
[,ShortRead, ANY, ANY-method (ShortRead-class), 40
[,ShortRead, ANY, missing, ANY-method (ShortRead-class), 40
[,ShortRead, ANY, missing-method (ShortRead-class), 40
[,ShortRead, missing, ANY, ANY-method (ShortRead-class), 40
[,ShortRead, missing, ANY-method (ShortRead-class), 40
[,ShortRead, missing, missing, ANY-method (ShortRead-class), 40
[,ShortRead, missing, missing-method (ShortRead-class), 40
[,ShortReadQ, ANY, ANY, ANY-method (ShortReadQ-class), 42
[,ShortReadQ, ANY, ANY-method (ShortReadQ-class), 42
[,ShortReadQ, ANY, missing, ANY-method (ShortReadQ-class), 42
[,ShortReadQ, missing, ANY-method (ShortReadQ-class), 42
[,ShortReadQ, missing, missing, ANY-method (ShortReadQ-class), 42
[,ShortReadQ, missing, missing-method (ShortReadQ-class), 42
[,SolexaIntensity, ANY, ANY, ANY-method (SolexaIntensity-class), 44

accessors, 1, 4, 40, 42
alignData (accessors), 1
alignDataFilter (srFilter), 55
AlignedDataFrame, 2, 3, 3
AlignedDataFrame-class, 2
AlignedRead, 2, 4, 5, 6, 24–27, 41
AlignedRead-class, 4
alignQuality (accessors), 1
alignQualityFilter (srFilter), 55
alphabet, FastqQuality-method (QualityScore-class), 21
alphabetByCycle, 6, 22, 41, 43
alphabetByCycle, BStringSet-method (alphabetByCycle), 6
alphabetByCycle, FastqQuality-method (QualityScore-class), 21
alphabetByCycle, ShortRead-method (ShortRead-class), 40
alphabetByCycle, ShortReadQ-method (ShortReadQ-class), 42
alphabetFrequency, 22
alphabetFrequency, FastqQuality-method (QualityScore-class), 21
alphabetScore, 8, 22, 43
alphabetScore, FastqQuality-method (QualityScore-class), 21
alphabetScore, SFastqQuality-method (QualityScore-class), 21
alphabetScore, ShortReadQ-method (ShortReadQ-class), 42
AlignedPath (accessors), 1
AnnotatedDataFrame, 2, 3, 45, 49
append, ShortReadBase, .ShortReadBase, ANY-method (ShortReadBase-package), 41
append, AlignedDataFrame, AlignedDataFrame, missing-method (AlignedDataFrame-class), 2
append, AlignedRead, AlignedRead, missing-method (AlignedRead-class), 4
append, MatrixQuality, MatrixQuality, missing-method (QualityScore-class), 21
append, QualityScore, QualityScore, missing-method (QualityScore-class), 21
append, ShortRead, ShortRead, missing-method (ShortRead-class), 40
append, ShortReadQ, ShortReadQ, missing-method (ShortReadQ-class), 42
ArrayIntensity (Intensity-class), 15
ArrayIntensity-class (Intensity-class), 15
baseCallPath (accessors), 1
basePath (deprecated), 11
BowtieQA, 36
BowtieQA-class, 8
BStringSet, 23
chromosome (accessors), 1
chromosomeFilter (srFilter), 55
clean, 9
clean, DNAStringSet-method (clean), 9
clean, ShortRead-method (ShortRead-class), 40
coerce, FastqQuality, matrix-method (QualityScore-class), 21
coerce, FastqQuality, numeric-method (QualityScore-class), 21
coerce, PairwiseAlignedXStringSet, AlignedRead-method (AlignedRead-class), 4
coerce, SFastqQuality, matrix-method (QualityScore-class), 21
compose (srFilter), 55
countLines, 10
coverage, AlignedRead-method (AlignedRead-class), 4
dataPath (accessors), 1
deprecated, 11
detail, 11
detail, ShortReadBase-method (SRUtil-class), 60
detail, AlignedRead-method (AlignedRead-class), 4
detail, ExperimentPath-method (ExperimentPath-class), 13
detail, QualityScore-method (QualityScore-class), 21
detail, RochePath-method (RochePath-class), 37
detail, ShortRead-method (ShortRead-class), 40
detail, ShortReadQ-method (ShortReadQ-class), 42
detail, SolexaPath-method (SolexaPath-class), 47
detail, SolexaSet-method (SolexaSet-class), 49
detail, SRList-method (SRUtil-class), 60
detail, SRSet-method (SRSet-class), 58
detail, SRVector-method (SRUtil-class), 60
dim, Intensity-method (Intensity-class), 15
dim, MatrixQuality-method (QualityScore-class), 21
dustyFilter (srFilter), 55
dustyScore, 12, 57
dustyScore, DNAStringSet-method (dustyScore), 12
dustyScore, ShortRead-method (dustyScore), 12
ExperimentPath, 38, 59
ExperimentPath (ExperimentPath-class), 13
experimentPath (accessors), 1
experimentPath, SRSet-method (SRSet-class), 58
ExperimentPath-class, 13
FastqQA, 20, 36
FastqQA (FastqQA-class), 14
FastqQA-class, 14
FastqQuality, 21
FastqQuality (QualityScore), 23
FastqQuality, BStringSet-method (QualityScore), 23
FastqQuality, character-method (QualityScore), 23
FastqQuality, missing-method (QualityScore), 23
FastqQuality-class (QualityScore-class), 21
function, 55
grep, 10, 24, 28, 30, 35, 56
id (accessors), 1
idFilter (srFilter), 55
imageAnalysisPath (accessors), 1
IntegerQuality (QualityScore), 23
IntegerQuality-class (QualityScore-class), 21
Intensity, 32, 44, 45
intensity (Intensity-class), 15
Intensity-class, 15
IntensityInfo, 44, 45
IntensityInfo-class (Intensity-class), 15
IntensityMeasure-class (Intensity-class), 15
is, 60

laneDescription (accessors), 1
laneNames (accessors), 1
laneNames, AnnotatedDataFrame-method (SolexaSet-class), 49
laneNames, SolexaSet-method (SolexaSet-class), 49
lapply, 61
lapply, SRList, ANY-method (SRUtil-class), 60
lapply, SRList-method (SRUtil-class), 60
length, MatrixQuality-method (QualityScore-class), 21
length, QualityScore-method (QualityScore-class), 21
length, ShortRead-method (ShortRead-class), 40
length, SRList-method (SRUtil-class), 60
list.files, 10

MAQMapQA, 20, 36
MAQMapQA (MAQMapQA-class), 16
MAQMapQA-class, 16
MatrixQuality (QualityScore), 23
MatrixQuality-class (QualityScore-class), 21
measurementError (Intensity-class), 15

name (SRFilter-class), 54
name, SRFilter-method (SRFilter-class), 54
names, SRList-method (SRUtil-class), 60
names<-, SRList, character-method (SRUtil-class), 60
narrow, 22, 40, 43

narrow, FastqQuality-method (QualityScore-class), 21
narrow, MatrixQuality-method (QualityScore-class), 21
narrow, ShortRead-method (ShortRead-class), 40
narrow, ShortReadQ-method (ShortReadQ-class), 42
nFilter (srFilter), 55
NumericQuality, 21, 22
NumericQuality (QualityScore), 23
NumericQuality-class (QualityScore-class), 21

OptionalFunction, 55

pairwiseAlignment, 52, 53
phenoData, 59
phenoData, SRSet-method (SRSet-class), 58
pileup, 17, 29
polyn (Utilities), 18
polynFilter (srFilter), 55
position (accessors), 1
positionFilter (srFilter), 55
PossibleMethod, 55

qa, 8, 9, 14, 16, 17, 19, 20, 36, 43, 44
qa, character-method (qa), 20
qa, SolexaPath-method (SolexaPath-class), 47
QualityScore, 8, 23, 23, 33
QualityScore-class, 21

qualPath (RochePath-class), 37
rank, 53

read454 (RochePath-class), 37
read454, RochePath-method (RochePath-class), 37
readAligned, 4, 5, 23, 24
readAligned, character-method (readAligned), 24
readAligned, SolexaPath-method (SolexaPath-class), 47
readAligned, SolexaSet-method (SolexaSet-class), 49
readBaseQuality, 28
readBaseQuality, character-method (readBaseQuality), 28
readBaseQuality, SolexaPath-method (SolexaPath-class), 47
readBfaToc, 17, 29
readCount (SRSet-class), 58
INDEX

readData, \texttt{RocheSet-class}, 38, 59
readFasta, \texttt{RochePath-class}, 37
readFasta, character-method (\texttt{RochePath-class}), 37
readFasta, \texttt{RochePath-method} (\texttt{RochePath-class}), 37
readFastq, 23, 30, 42, 43
readFastq, character-method (readFastq), 30
readFastq, \texttt{SolexaPath-method} (\texttt{SolexaPath-class}), 47
readIndex, \texttt{RocheSet-class}, 58
readInfo, \texttt{Intensity-class}, 15
readIntensities, 15, 16, 32, 45, 46
readIntensities, character-method (readIntensities), 32
readIntensities, \texttt{SolexaPath-method} (\texttt{SolexaPath-class}), 47
readPath, \texttt{RochePath-class}, 37
readPrb, 29, 33
readPrb, character-method, 48
readPrb, character-method (readPrb), 33
readQseq, \texttt{SolexaPath-method} (\texttt{SolexaPath-class}), 47
readQseq, \texttt{FastqQuality-method} (\texttt{QualityScore-class}), 23
readQseq, \texttt{missing-method} (\texttt{QualityScore-class}), 23
readQseq, \texttt{missing-method} (\texttt{QualityScore-class}), 23
readQual, \texttt{RochePath-class}, 37
readQual, character-method (\texttt{RochePath-class}), 37
readQual, \texttt{RochePath-method} (\texttt{RochePath-class}), 37
readXStringColumns, 25, 29, 35
report, 20, 36
report, \texttt{ANY-method} (report), 36
report, \texttt{BowtieQA-method} (\texttt{BowtieQA-class}), 8
report, \texttt{FastqQA-method} (\texttt{FastqQA-class}), 14
report, \texttt{MAQMapQA-method} (\texttt{MAQMapQA-class}), 16
report, \texttt{SolexaExportQA-method} (\texttt{SolexaExportQA-class}), 43
report, \texttt{SolexaPath-method} (\texttt{SolexaPath-class}), 47
RochePath, 37
RochePath, \texttt{RochePath-class}, 37
RochePath-class, 37
sapply, 61
sapply, \texttt{SRList-method} (\texttt{SRUtil-class}), 60
scanPath (accessors), 1
\texttt{SFastqQuality}, 48
\texttt{SFastqQuality} (\texttt{QualityScore-class}), 23
\texttt{SFastqQuality} (\texttt{BStringSet-method} (\texttt{QualityScore-class}), 23
\texttt{SFastqQuality}, character-method (\texttt{QualityScore-class}), 23
\texttt{SFastqQuality}, missing-method (\texttt{QualityScore-class}), 23
\texttt{ShortRead}, 4, 42, 54
\texttt{ShortRead} (\texttt{ShortRead-class}), 40
\texttt{ShortRead}, \texttt{DNAStringSet,BStringSet-method} (\texttt{ShortRead-class}), 40
\texttt{ShortRead}, \texttt{DNAStringSet,missing-method} (\texttt{ShortRead-class}), 40
\texttt{ShortRead}, 40
\texttt{ShortReadBase-package}, 41
\texttt{ShortReadQ}, 2, 4, 11, 28, 29, 31, 34, 38, 41, 48
\texttt{ShortReadQ} (\texttt{ShortReadQ-class}), 42
\texttt{ShortReadQ}, \texttt{DNAStringSet,QualityScore,BStringSet-method} (\texttt{ShortReadQ-class}), 42
\texttt{ShortReadQ}, \texttt{DNAStringSet,missing-method} (\texttt{ShortReadQ-class}), 42
\texttt{ShortReadQ}, \texttt{missing,missing,missing-method} (\texttt{ShortReadQ-class}), 42
\texttt{show}, 11
\texttt{show}, \texttt{.QA-method} (\texttt{SolexaExportQA-class}), 43
\texttt{show}, \texttt{.ShortReadBase-method} (\texttt{ShortReadBase-package}), 41
\texttt{show}, \texttt{AlignedRead-method} (\texttt{AlignedRead-class}), 4
INDEX

srrank,XStringSet-method (srduplicated), 53
SRSet, 39
SRSet-class, 58
srsort, 22
srsort (srduplicated), 53
srsort,FastqQuality-method (QualityScore-class), 21
srsort,ShortRead-method (ShortRead-class), 40
srsort,XStringSet-method (srduplicated), 53
SRUtil-class, 60
SRVector(SRUtil-class), 60
SRVector-class (SRUtil-class), 60
SRWarn(SRUtil-class), 60
SRWarn-class (SRUtil-class), 60
strand,AlignedRead-method (AlignedRead-class), 4
strandFilter (srFilter), 55
tables, 41, 62
tables,ShortRead-method (ShortRead-class), 40
tables,XStringSet-method (tables), 62
trimLRPatterns, 41
trimLRPatterns,ShortRead-method (ShortRead-class), 40

uniqueFilter (srFilter), 55
Utilities, 18

vclass (accessors), 1

Versioned, 3, 45

width,FastqQuality-method (QualityScore-class), 21
width,MatrixQuality-method (QualityScore-class), 21
width,NumericQuality-method (QualityScore-class), 21
width,QualityScore-method (QualityScore-class), 21
width,ShortRead-method (ShortRead-class), 40
writeFastq (readFastq), 30
writeFastq,ShortReadQ-method (ShortReadQ-class), 42

XStringSet, 7, 54, 62