Package ‘spliceSites’

March 23, 2017

Type       Package
Title      A bioconductor package for exploration of alignment gap
           positions from RNA-seq data
Version    1.22.1
Date       2012-10-28
Author     Wolfgang Kaisers
Maintainer Wolfgang Kaisers <kaisers@med.uni-duesseldorf.de>
Description Performs splice centered analysis on RNA-seq data.
License    GPL-2
biocViews  RNAseq,GeneExpression,DifferentialExpression,Proteomics
Depends    methods,rbamtools (>= 2.14.3),refGenome (>= 1.6.0),Biobase,Biostrings (>= 2.28.0)
Imports    BiocGenerics,doBy,seqLogo,IRanges
Collate    allClasses.r allGenerics.r c-methods.r dim-methods.r
           head-methods.r show-methods.r spliceSites.r
NeedsCompilation yes

R topics documented:

spliceSites-package .................................................. 2
aaGapSites-class ...................................................... 3
addGeneAligns ........................................................ 5
addHbond ............................................................... 6
addMaxEnt ............................................................... 7
alt_X_ranks ........................................................... 8
annGapSites-class ..................................................... 9
annotate-ExpressionSet .............................................. 10
annotation ............................................................ 11
as.data.frame-methods .............................................. 12
c-methods ............................................................. 12
caRanges-class ....................................................... 13
cdRanges-class ....................................................... 14
countByGeneName ..................................................... 15
cRanges-class ........................................................ 16
dim-methods .......................................................... 18
dnaGapSites-class ..................................................... 18
spliceSites-package

Calculate information on splice-sites from gapped alignments in RNA-seq data.

Description

The package defines 'cRanges' the (centered ranges) class which represents a genomic range that contains a highlighted position (center): This will usually be the boundary between an exon and an intron. The second defined type is the class 'gapSites' which represents two exonic regions divided by a gap (usually an intron). There are subclasses which additionally contain DNA or AA sequences.
Details

Package: spliceSites
Type: Package
Version: 1.0
Date: 2012-10-28
License: GPL-2
Depends: methods, rbamtools, refGenome, Biobase, BiocGenerics, Biostrings, seqLogo

Author(s)

Wolfgang Kaisers
Maintainer: Wolfgang Kaisers <kaisers@med.uni-duesseldorf.de>

References

http://genes.mit.edu/burgelab/maxent/Xmaxentscan_scoreseq.html

See Also

rbamtools refGenome

Examples

bam <- system.file("extdata", "rna_fem.bam", package="spliceSites")
reader <- bamReader(bam, idx=TRUE)
 ga <- alignGapList(reader)
bamClose(reader)
dnafile <- system.file("extdata", "dna_small.RData", package="spliceSites")
load(dnafile)
ucf <- system.file("extdata", "uc_small_junc.RData", package="spliceSites")
ucj <- loadGenome(ucf)
annotation(ga) <- annotate(ga, ucj)
ga

Description

Contains gapAligns data and a AAStringSet.

Objects from the Class

Objects can be created by calls of the form new("aaGapSites", ...).
Slots

seq: "AStringSet": Contains amino acid sequences.
nAligns: "numeric": Contains total number of aligns.
nAlignGaps: "numeric": Contains total number of align gaps.
dt: "data.frame": Contains data for all gap sites.

Extends

Class "gapSites", directly.

Methods

head signature(x = "aaGapSites"): Returns the first lines of object.
show signature(object = "aaGapSites"): Returns the last lines of object.
truncatoseq signature(x="caRanges",rme=TRUE, trunc=42L): Truncates contained sequence when character (given by ASCII code in trunc). The default (42L) encodes for character ‘*’ which indicates stop-codon.
trypsinCleave signature(x = "caRanges", minLen = 5): Performs in silico trypsinization of contained sequence. The sequence fragment which contains the (position depicted) exon-intron boundary is returned. Datasets for which the truncated sequence is shorter than minLen are excluded.
write.files signature(x = "caRanges"):Exports contained data into "csv" file.

Author(s)

Wolfgang Kaisers

Examples

# A) Read gap-sites from BAM-file
bam <- system.file("extdata", "rna_fem.bam", package="spliceSites")
reader <- bamReader(bam, idx=TRUE)
ga <- alignGapList(reader)
bamClose(reader)

# B) Load reference dna
dnafile <- system.file("extdata", "dna_small.RData", package="spliceSites")
load(dnafile)

# C) Calculate cross junctional ranges
lrc <- lrJunc(ga, lfeatlen=6, rfeatlen=6, strand=’+’)
lr1 <- lrCodons(lrc, frame=1, strand=’+’)
lr2 <- lrCodons(lrc, frame=2, strand=’+’)
lr3 <- lrCodons(lrc, frame=3, strand=’+’)
lr <- c(lr1, lr2, lr3)

# D) Add DNA-sequence
lrd <- dnaGapSites(lr, dna_small)

# E) Translate DNA to amino acid
lra <- translate(lrd)
addGeneAligns

Description

Locates gene in genome via refGenome and reads a bamRange from the determined region.

Usage

addGeneAligns(x)

Arguments

x  gapSites. The result contains a copy of the passed object.

Details

The function adds a gene_aligns column to the contained data.frame.

Value

gapSites

Author(s)

Wolfgang Kaisers

Examples

# A) Read gapSites
bam <- system.file("extdata", "rna_fem.bam", package="spliceSites")
reader <- bamReader(bam, idx=TRUE)
ga <- alignGapList(reader)
bamClose(reader)

# B) Annotate
ucf <- system.file("extdata", "uc_small_junc.RData", package="spliceSites")
ucj <- loadGenome(ucf)
annotation(ga) <- annotate(ga, ucj)

# C) align part
gal <- addGeneAligns(ga)
gal
addHbond

Class "hbond": Provides data and functions for calculation of HBond scores for 5’ splice-sites.

Description

The addHbond methods add HBond scores to gapSites and cdRanges objects. HBond scores provide a measure for the capability of a 5’ splice-site to form H-bonds with the U1 snRNA. The function requires at least 3 exon nucleotides and 8 intron nucleotides. The first two intron nucleotides are expected to be ’GT’ (for other values the returned score will be 0). The routine equally accepts upper and lower case characters.

Usage

addHbond(x,dna)

Arguments

x          gapSites. The object to which HBond scores are added.
dna        DNAStringSet. Reference sequence identifier.

Details

In cdRanges objects, the function adds a hbond column. In gapSites objects, the function adds a lhbond (left side) and a rhbond (right side) column. The lhbond values always assume ‘+’-strand (because HBond works on the 5’ side). The rhbond values always assume ‘-’-strand. Therefore, there will be discrepancies in the output of write.annDNA.tables because the leftseq and rightseq sequences are reverse-complemented according to the strand column: The xhbond may be > 0 without GT at position 4 (but with AC at position 7).

Author(s)

Wolfgang Kaisers

References

http://www.uni-duesseldorf.de/rna/html/hbond_score.php

Examples

# A) Read gapSites
bam<-system.file("extdata","rna_fem.bam",package="spliceSites")
reader<-bamReader(bam,idx=TRUE)
ga<-alignGapList(reader)
bamClose(reader)

# B) Load DNA
dnafile<-system.file("extdata","dna_small.RData",package="spliceSites")
load(dnafile)

# C) HBond
gab<-addHbond(ga,dna_small)
addMaxEnt

# D) cdRanges
lj<-lJunc(ga, featlen=3, gaplen=8, strand=’+’)
ljd<-dnaRanges(lj, dna_small)
ljdh<-addHbond(ljd)

addMaxEnt: Extract subset of data contained in given range given object.

Description
addMaxEnt adds new columns to object data which contain MaxEnt-Score derived values. mxe_ps5 contains score5 values for left align-gap (exon-intron) boundary (i.e. assumed to reside on ' + '-strand. mxe_ps3 contains score3 (maxent) values for right align-gap (intron-exon) boundary (i.e. assumed to reside on ' + '-strand).
mxe_ms5 contains score5 values for right align-gap (exon-intron) boundary on reverseComplement transformed sequence (i.e. assumed to reside on ' - '-strand).
mxe_ms3 contains score3 values for left align-gap (intron-exon) boundary on reverseComplement transformed sequence (i.e. assumed to reside on ' - '-strand).

From these values, s3strand, s5strand and meStrand are derived: s3strand is ' + ' when mxe_ps5 >= mxe_ms5 and ' - ' otherwise; s3strand is ' + ' when mxe_ps3 >= mxe_ms3 and ' - ' otherwise.
meStrand equals s5strand when s5strand=s3strand and '*' otherwise.

The function setMeStrand copies existing meStrand values into strand column (and throws an error when meStrand does not exist).

Usage
addMaxEnt(x, dna, maxent, digits=1)

Arguments
x gapSites.
dna DNAStringSet. Reference sequence identifier.
maxent maxEnt. Contains score table which are internally used by score3 and score5 methods.
digits Numeric. Default value: 1. Internally calculated maxent scores are rounded to given number of decimal places.

Value
gapSites

Author(s)
Wolfgang Kaisers
Examples

A) Read gapSites

```r
bam<-'system.file("extdata","rna_fem.bam",package="spliceSites")
reader<-'bamReader(bam,idx=TRUE)
ga<-alignGapList(reader)
bamClose(reader)
```

B) Load DNA

```r
dnafile<-'system.file("extdata","dna_small.RData",package="spliceSites")
load(dnafile)
```

C) maxEnt

```r
mes<-'load.maxEnt()
gae<-'addMaxEnt(ga,dna_small,mes)
getMeStrand(gae)
sae<-'setMeStrand(gae)
```

---

**alt_X_ranks**

*alt_left_ranks and alt_right_ranks functions: Identification of alternative splicing events from gapped alignments.*

---

**Description**

`alt_X_ranks` covers the functions `alt_left_ranks` and `alt_right_ranks`. Both functions identify alternative splice-sites. `alt_left_ranks` finds sites which share the same `rstart` value (on the same `seqid`). `alt_right_ranks` finds sites which share the same `lend` value (on the same `seqid`). `alt_ranks` combines the results of both functions together with `seqid`, `lend` and `rstart` values in one table.

**Usage**

```r
alt_left_ranks(x)
```

**Arguments**

- `x` gapSites. Object for which alternative ranks are calculated

**Details**

The function `alt_left_ranks` groups align-gaps (splice-sites) which share identical `rstart` position and have different `lend` position. Each Group is assigned a unique `alt_id` (integer value beginning from 1). The first column in the returned data.frame is an id-column which facilitates table merging with the source table. The result has the same number of rows as the source and the id-column.

**Value**

`data.frame`. The table contains the columns `nr_alt`, `alt_id`, `id`, `diff_ranks` and `gap_diff`.

**Author(s)**

Wolfgang Kaisers
Examples

# A) Read gapSites
bam<-system.file("extdata","rna_fem.bam",package="spliceSites")
reader<-bamReader(bam,idx=TRUE)
ga<-alignGapList(reader)
bamClose(reader)

# B) alt_ranks
alr<-alt_left_ranks(ga)
ar<-alt_ranks(ga)

Description

Contains data from align gaps together with annotation data (and optional data about alternative splice positions). Objects of this class are returned from the annotation member function for class gapSites.

Details

plot_diff plots tabulated distance between inner gap-site border and annotated exon-intron boundaries.

Objects from the Class

Objects can be created by calls of the form annotation on gapSites objects.

Slots

nAligns: Object of class "numeric" Total number of aligns.
nAlignGaps: Object of class "numeric" Total number of gapped aligns.
dt: "data.frame". Contains gap-positions, annotation data and optional alternative position data.
annotation: "data.frame". Contains annotation data.
profile: "data.frame". Contains descriptive data for source probes (BAM-files).

Extends

Class "gapSites", directly.

Methods

as.data.frame signature(x = "annGapSites"): Returns the contained data.

Author(s)

Wolfgang Kaisers
annotate-ExpressionSet

Examples

# A) Read gapSites from BAM
bam <- system.file("extdata", "rna_fem.bam", package="spliceSites")
reader <- bamReader(bam, idx=TRUE)
ga <- alignGapList(reader)
bamClose(reader)

# B) Load annotation data
ucf <- system.file("extdata", "uc_small_junc.RData", package="spliceSites")
ucj <- loadGenome(ucf)

# C) Add Annotation
annotation(ga) <- annotate(ga, ucj)

# D) Retrieve annotation
aga <- annotation(ga)
aga

# D) plot_diff
aga <- annotation(ga)
plot_diff(aga)

annotate-ExpressionSet

Adds annotation data to existing ExpressionSet (created by readExpSet)

Description

 Reads featureData from incoming Expression set which should contain range data on embedding exons for gap-sites. The annotate function then overlaps the ranges with given annotation data. The result of overlapping is written into a AnnotatedDataFrame.

Arguments

object       ExpressionSet
genome       refGenome

Value

AnnotatedDataFrame

Author(s)

Wolfgang Kaisers

Examples

# A) Names of BAM-files
bam <- character(2)
bam[1] <- system.file("extdata", "rna_fem.bam", package="spliceSites")
# B) Experiment Profile
prof <- data.frame(gender=c("f", "m"))
meta <- data.frame(labelDescription=names(prof), row.names=names(prof))
pd<-new("AnnotatedDataFrame", data=prof, varMetadata=meta)

# C) Read ExpressionSet
es <- readExpSet(bam, phenoData=pd)

# D) Annotate ExpressionSet
ucf <- system.file("extdata", "uc_small.RData", package="spliceSites")
uc <- loadGenome(ucf)
juc <- getSpliceTable(uc)
ann <- annotate(es, juc)

---

**annotation**

Annotation functions for **gapSites** objects

**Description**

The `annotate` function takes a `gapSites` and a `refGenome` object and returns a list which additionally contains a ‘class’ attribute ‘annotationResult’. The object is intended as input for the `annotation` member function of class `gapSites`. The `annotation` member functions act as writing and reading accessor for annotation data inside `gapSites` objects.

**Usage**

`annotate(object, junc)`

**Arguments**

- `object` [gapSites]. Align-gap data for which annotations are provided via overlap.
- `junc` [refJunctions]. Object which provides annotated splice site positions.

**Details**

The `annotation` reading accessor takes a `gapSites` object and returns an `annAlignGaps` object. The `annotation` writing accessor takes a `gapSites` and an `annotationResult` object and copies the contained table into the annotation slot of the `gapSites` object.

**Value**

`annAlignGaps`

**Author(s)**

Wolfgang Kaisers
Examples

# A) Create gapSites object
bam <- system.file("extdata", "rna_fem.bam", package="spliceSites")
reader <- bamReader(bam[1], idx=TRUE)
ga <- alignGapList(reader)
bamClose(reader)

# B) Read refGenome object
ucf <- system.file("extdata", "uc_small_junc.RData", package="spliceSites")
ucj <- loadGenome(ucf)

# C) Add annotation data
annotation(ga) <- annotate(ga, ucj)

as.data.frame-methods

Description

Methods for function as.data.frame

Methods

signature(x = "gapSites") Method for 'gapSites'.
signature(x = "annGapSites") Method for 'annGapSites'.

C-methods

Description

Coercing functions c.

Methods

signature(x = "cRanges") Method for 'cRanges'.
signature(x = "gapSites") Method for 'gapSites'.
Description

"caRanges" Objects that contain a centered genomic range and amino acid sequences.

Objects from the Class

Objects are usually created from objects of class "cdRanges" by the "translate" function.

Slots

dt: Object of class "data.frame". Contains the columns "seqid", "start", "end", "strand", "position", "id", "frame".

seq: Object of class "AAStringSet". Contains amino-acid-sequence of ranges described in dt.

Extends

Class "cRanges", directly.

Methods

c signature(x = "caRanges"): Generic combining for caRanges objects.

getSequence signature(x = "caRanges"): Returns contained sequence (DNAStringSet).

head signature(x = "aaGapAligns"): Returns the first lines of object.

show signature(object = "aaGapAligns"): Returns the last lines of object.

truncateSeq signature(x = "caRanges", rme = TRUE, trunc = 42L): Truncates contained sequence when character (given by ASCII code in trunc). The default (42L) encodes for character '*' which indicates stop-codon.

trypsinCleave signature(x = "caRanges", minLen = 5): Performs in silico trypsinization of contained sequence. The sequence fragment which contains the (position depicted) exon-intron boundary is returned. Datasets for which the truncated sequence is shorter than minLen are excluded.

write.files signature(x = "caRanges"): Exports contained data into "csv" file.

Author(s)

Wolfgang Kaisers

See Also

cRanges
Examples

# A) Read gapSites object
bam<-system.file("extdata","rna_fem.bam",package="spliceSites")
reader<-bamReader(bam,idx=TRUE)
ga<-alignGapList(reader)
bamClose(reader)
ga

# B) Create cRanges object
lj<-lJunc(ga,featlen=21,gaplen=21,strand="+")
ljc$lCodons(lj,frame=1,keepStrand=TRUE)
dnafile<-system.file("extdata","dna_small.RData",package="spliceSites")
load(dnafile)
# C) Add DNA sequence
cdr<-$dnaRanges(ljc,dna_small)
# D) Translate into AA sequence
ar<-$translate(cdr)
# E) Truncate and cleave...
trc<-$trypsinCleave(trc)

cdRanges-class

Class "cdRanges"

Description

"cdRanges" Objects that contain centered Ranges (exon-intron junctions) and dna-sequences.

Objects from the Class

Objects are usually created from "cRanges" with the function "dnaRanges".

Slots

dt: Object of class "data.frame". Contains the columns "seqid","start","end","strand","position","id","frame"

seq: Object of class "DNAStringSet". Contains the dna-sequence of ranges described in dt.

Extends

Class "cRanges", directly.

Methods

c signature(x = "cdRanges"): Generic combining for cdRanges objects.

getSequence signature(x="cdRanges"): Returns contained sequence (DNAStringSet).

head signature(x = "cdRanges"): Prints first items from object.

initialize signature(.Object = "cdRanges"): Create an instance of class using new.

seqlogo signature(x = "cdRanges"): Show a seqlogo of contained sequences

translate signature(x = "cdRanges"): Translates dna-sequence into amino-acid-sequence. Returns an object of class "caRanges").
countByGeneName

Author(s)

Wolfgang Kaisers

See Also

cRanges

Examples

# A) Read gapSites object
bam<-system.file("extdata","rna_fem.bam",package="spliceSites")
reader<-bamReader(bam,idx=TRUE)
ga<-alignGapList(reader)
bamClose(reader)
ga
# B) Create cRanges object
lj<--lJunc(ga,featlen=21,gaplen=21,strand="+"
ljc<--lCodons(lj,frame=1,keepStrand=TRUE)
dnafile<--system.file("extdata","dna_small.RData",package="spliceSites")
load(dnafile)
# C) Add DNA sequence
cdr<-dnaRanges(ljc,dna_small)
# D) seqLogo ...
seqlogo(cdr)

countByGeneName

Description

Opens multiple BAM-files and reads aligns for selected gene for each file. The function counts the
tag-selected value which either is a BAM-cigar operation (like "N" or "M") or the total number of
aligns.

Usage

countByGeneName(object,infiles,idxInfiles=paste(infiles,".bai",sep=""),gene,tag="N")

Arguments

object Object of class "refGenome"
infiles Vector of BAM-files
idxInfiles (Optional) Vector of BAM-index files.
gene Gene name
tag Character. Passed to (rbamtools) 'bamCountAll' function. Default value is "N". Other accepted values include "nAligns","M","I","D". 
Details

countByGeneName first uses the extractByGeneName and getGenePositions from 'refGenome' in order to calculate coordinates from the given gene name. Then for each given BAM-file name, the functions calls the bamCount function and returns a vector with a count value for each given file. Internally countByGeneName also checks for existing BAM-index file and tries to create index files which do not exist.

Value

Numeric vector. Length equals number of BAM-input files.

Author(s)

Wolfgang Kaisers

Examples

# A) Read filenames
ucf<-system.file("extdata", "uc_small.RData", package="spliceSites")
uc<-loadGenome(ucf)
bam<-character(2)
bam[1]<-system.file("extdata", "rna_fem.bam", package="spliceSites")
# B) count
countByGeneName(uc, bam, gene="WASH7P", tag="N")
countByGeneName(uc, bam, gene="WASH7P", tag="nAligns")

---
cRanges-class

Class "cRanges": Centered ranges.

Description

"cRanges" Objects that contain centered genomic ranges. The center position marks a prominent position inside the range, generally an exon-intron junction. Position values represent the 0-based position of last exon nucleotide.

Objects from the Class

Objects can be created by calls of the form new("cRanges", seqnames, start, end, width, strand, position, id)

Methods

as.data.frame signature(x = "cRanges"): Returns a copy of the contained data inside a data.frame object.
c signature(x = "cRanges"): Generic combining for cRanges objects.
count signature(x = "cRanges"): Returns the number of contained ranges (number of rows).
cRanges-class

**dim** signature(x = "cRanges"): Returns the dim of the contained data.frame.

**dnaRanges** signature(x = "cRanges", dname="DNAStringSet",useStrand="logical",removeUnknownStrand=logical): Takes a cRanges object and a DNAStringSet (a reference sequence) and adds the appropriate DNA sequence to the genomic ranges. Returns a cdRanges object.

**end** signature(x = "cRanges"): Returns end column of data.

**head** signature(x = "cRanges",n="numeric",digits="numeric"): Returns first n (default: n=6) lines of contained data.frame.

**id** signature(x = "cRanges"): Returns id column from contained data.frame.

**initialize** signature(.Object = "cRanges"): Generic class initialisation method.

**lCodons** signature(x = "cRanges",frame="numeric", keepStrand="logical"): Returns cRanges object which represents ranges truncated to codon size. When 'keepStrand' is set to FALSE, strand is set to '+'. The intention is that appended DNA sequences which then can be translated into amino acids.

**rCodons** signature(x = "cRanges",frame="numeric", keepStrand="logical"): Returns cRanges object which represents ranges truncated to codon size. When 'keepStrand' is set to FALSE, strand is set to '+'. The intention is that appended DNA sequences which then can be translated into amino acids.

**seqid** signature(x = "cRanges"): Returns vector with seqid's.

**show** signature(object = "cRanges"): Generic print function.

**sortTable** signature(x="cRanges"): Sort contained tables by seqid, lend and rstart.

**start** signature(x = "cRanges"): Returns start column from contained data.frame.

**strand** signature(x = "cRanges"): Returns strand column from contained data.frame.

**width** signature(x = "cRanges"): Returns width of contained ranges (=end-start+1).

**Author(s)**

Wolfgang Kaisers

**See Also**

gapRanges

**Examples**

# A) Create cRanges object from scratch
sq<-factor(c(1,1,2,2,3,3),labels=c("chr1","chr2","chr3"))
st<-c(100,200,100,300,100,400)
en<-c(120,210,110,310,110,410)
pos<-c(2,3,4,5,6,7)

# Create cRanges
cr<-new("cRanges",seqid=sq,start=st,end=en,position=pos)

# Print cRanges
print(cr)

# Print id of cRanges
print(id(cr))

# Sort cRanges
print(sortTable(cr))

# Return end column of cRanges
print(end(cr))

# Return start column of cRanges
print(start(cr))

# Return strand column of cRanges
print(strand(cr))

# Return width of cRanges
print(width(cr))

# Return id of cRanges
print(id(cr))

# Print lCodons of cRanges with frame 1 and keepStrand=TRUE
print(lCodons(cr,frame=1,keepStrand=TRUE))

# Print lCodons of cRanges with frame 1 and keepStrand=FALSE
print(lCodons(cr,frame=1,keepStrand=FALSE))

# Print lCodons of cRanges with frame 2 and keepStrand=TRUE
print(lCodons(cr,frame=2,keepStrand=TRUE))
dnaGapSites-class

rCodons(cr,frame=1,keepStrand=FALSE)
# + + + + + + + + + + + + + + + + + + + + + + + + + + + + + +
# B) Intended way to create a cRanges object from BAM data
bam<-system.file("extdata","rna_fem.bam",package="spliceSites")
reader<-bamReader(bam,idx=TRUE)
ga<-alignGapList(reader)
bamClose(reader)
ga
lj<-lJunc(ga,featlen=3,gaplen=6,strand="+")
lj
# C) ...
table(strand(lj))

---

dim-methods
dim: Return dimensions of contained data.frame.

Description
Methods for function dim

Methods
signature(x = "cRanges") Method for 'cRanges'.
signature(x = "gapSites") Method for 'gapSites'.

---

dnaGapSites-class
Class "dnaGapSites"

Description
dnaGapSites contains all data which is stored in objects of class "gapSites" plus additional DNA sequences in the "seq" slot.

Objects from the Class
Objects are usually created from gapSites via dnaGapSites.

Slots
seq: "DNAStringSet". Contains DNA sequence.
nAligns: code"numeric". Contains total number of aligns.
nAlignGaps: "numeric". Contains total number of align gaps.
dt: code"data.frame". Contains data on gap-sites.

Extends
Class "gapSites", directly.
dnaRanges

Methods

head signature(x = "dnaGapSites"): Returns head of dt.

seqlogo signature(x = "dnaGapSites"): Prints seq-logo of stored dna-sequence.

show signature(object = "dnaGapSites"): Prints head of dt.

translate signature(x = "dnaGapSites"): Returns an object of class aaalignGaps by translating seq into amino acids.

Author(s)

Wolfgang Kaisers

See Also

gapSites

Examples

# A) Read gapSites
bam<-system.file("extdata", "rna_fem.bam", package="spliceSites")
reader<-bamReader(bam, idx=TRUE)
ga<-alignGapList(reader)
# B) Load DNA sequence
dnafile<-system.file("extdata","dna_small.RData",package="spliceSites")
load(dnafile)
# C 1) Add DNA
dga<-dnaGapSites(ga,dna_small)
dga
# C 2) Calculate codon positions
lrj<-lrJunc(ga,lfeatlen=6,rfeatlen=6,strand="+")
lrc<-lrCodons(lrj,frame=1,strand="+")
# D) Add DNA sequence and translate
lrd<-dnaGapSites(lrc,dna_small)
lra<-translate(lrd)
lra

dnaRanges (Reads a bamRange object for a given bamReader, refGenome and gene name.)

Description

Locates gene in genome via refGenome and reads a bamRange from the determined region.

Usage

dnaRanges(x,dnaset,useStrand=TRUE,removeUnknownStrand=TRUE,verbose=TRUE,...)
Arguments

\begin{itemize}
\item \texttt{x}\hspace{1cm}cRanges. Range-data will be copied from this object.
\item \texttt{dnaset}\hspace{1cm}DNAStringSet. Contains the reference sequence from which the DNA-sequence is extracted.
\item \texttt{useStrand}\hspace{1cm}logical. When TRUE, sequences for which \texttt{strand}='-' are reverse-complemented.
\item \texttt{removeUnknownStrand}\hspace{1cm}logical. When TRUE, sequences for which \texttt{strand}='-' are removed.
\item \texttt{verbose}\hspace{1cm}logical. Determines amount of console output during routine runtime.
\item \texttt{...}\hspace{1cm}Optional additional arguments (currently unused).
\end{itemize}

Value

cdRanges

Author(s)

Wolfgang Kaisers

Examples

\begin{verbatim}
bam<-system.file("extdata","rna_fem.bam",package="spliceSites")
reader<-bamReader(bam,idx=TRUE)
gamalignGapList(reader)
bamClose(reader)
lj<-lJunc(ga,featlen=6,gaplen=6,strand="+")
dnafile<-system.file("extdata","dna_small.RData",package="spliceSites")
load(dnafile)
ljd<-dnaRanges(lj,dna_small)
seqlogo(ljd)
\end{verbatim}

\begin{verbatim}
extractByGeneName
\end{verbatim}

\textit{extractByGeneName: Extract subset for sites which lie in range(s) defined by given gene list.}

Description

The function takes a `cRanges` object (or derived) and searches inside of given `refGenome` object for gene names. From identified gene-name matches genomic target regions can be defined for which in turn the contained sites are extracted.

Usage

\texttt{extractByGeneName(object,geneNames,src,...)}

Arguments

\begin{itemize}
\item \texttt{object}\hspace{1cm}gapSites or cRanges (or derived). Object inside which the data is searched for.
\item \texttt{geneNames}\hspace{1cm}Character. Vector of gene names for which data is to be extracted.
\item \texttt{src}\hspace{1cm}refGenome. Contains gene annotation (for conversion of gene-name to genomic coordinates).
\item \texttt{...}\hspace{1cm}(currently unused)
\end{itemize}
extractRange

**Details**

The function internally calls `extractByGeneName` on `refGenome`. This function also prints out non matching gene names. On the result, the function calls `getGenePositions` from which the genomic regions can be extracted. For each gene, data is extracted via `extractRange` and the resulting objects are then concatenated.

**Value**

Same type as object

**Author(s)**

Wolfgang Kaisers

**Examples**

```r
# A) Read gapSites from BAM
bam<-'system.file("extdata","rna_fem.bam",package="spliceSites")
reader<-'bamReader(bam,idx=TRUE)
ga<-'alignGapList(reader)
bamClose(reader)
# B) Load DNAStringSet
dnafile<-'system.file("extdata","dna_small.RData",package="spliceSites")
load(dnafile)
# C) load refGenome
ucf<-'system.file("extdata","uc_small.RData",package="spliceSites")
uc<-'loadGenome(ucf)
# D) For cRanges
lj<-'lJunc(ga,featlen=6,gaplen=3,strand='+')
ljw<-'extractByGeneName(lj,geneNames="WASH7P",src=uc)
# E) For cdRanges
ljc<-'lCodons(lj,frame=2)
ljcd<-'dnaRanges(ljc,dna_small)
ljcdw<-'extractByGeneName(ljcd,geneNames="WASH7P",src=uc)
# F) For caRanges
ljca<-'translate(ljcd)
ljca<-'extractByGeneName(ljca,geneNames="WASH7P",src=uc)
# G) For gapSites
lrj<-'lrJunc(ga,lfeatlen=6,rfeatlen=6,strand='+')
lrw<-'extractByGeneName(lrj,geneNames="WASH7P",src=uc)
```

**Usage**

```r
extractRange(object, seqid, start, end)
```

**extractRange**

`extractRange`: Extract subset from object where records lie in given range.

**Description**

Searches in object for data which lie inside the given range and returns an object of same type containing extracted data.
Arguments

- **object**: gapSites or cRanges (or derived). Object inside which the data is searched for.
- **seqid**: character. Reference sequence identifier.
- **start**: numeric. Start position of given range.
- **end**: numeric. End position of given range.

Value

Same type as object

Author(s)

Wolfgang Kaisers

Examples

```r
# A) Read gapSites
bam<-system.file("extdata","rna_fem.bam",package="spliceSites")
reader<-bamReader(bam,idx=TRUE)
ga<-alignGapList(reader)
bamClose(reader)
# B) Load refGenome
ucf<-system.file("extdata","uc_small.RData",package="spliceSites")
uc<-loadGenome(ucf)
# C) For gapSites
extractRange(ga,seqid="chr1",start=14000,end=30000)
# D) For cRanges
lj<lJunc(ga,featlen=3,gaplen=6,strand="+")
extractRange(lj,seqid="chr1",start=14000,end=30000)
```

gapSites

Creating 'gapSites' and 'dnaGapSites' objects.

description

gapSites creates objects of class gapSites from scratch. dnaGapSites creates objects of class dnaGapSites from gapSites objects.

Usage

```r
gapSites(seqid=Factor(),lstart=integer(),lend=integer(),
        rstart=integer(),rend=integer(),gaplen,strand,
        nr_aligns=1,nAligns=sum(nr_aligns),
        nAlignGaps=sum(nr_aligns),nProbes=1)
```
Arguments

seqid  Character. Identifies reference sequence.
lstart Coordinates for start of left range.
lend  Coordinates for end of left range. Usually exon-intron boundary.
rstart Coordinates for start of right range. Usually exon-intron boundary.
rend  Coordinates for end of right range.
gaplen Length of enclosed gap. Should equal rstart-lend-1.
strand  + or - or * (for unknown). Default: '*'.
nr_aligns Number of gapped aligns which have the same exon-intron boundaries (lend and rstart)
nAligns Total number of aligns for probeset.
nAlignGaps Total number of gapped aligns for probeset.
nProbes Numeric. Number of probes in which this gapped position is present.

Details

The intended way to create a gapSites object is to use the alignGapList function which in turn calls the (rbamtools) bamGapList function. When a BAM file almost exclusively contains gapped aligns which sometimes are multiply gapped, possibly the 'nAlignGaps' value is greater than the 'nAligns'. When reading BAM files which contain the complete date of an alignment, usually the 'nAlignGaps' value is about $1/3$ of the 'nAligns' value.

Value

An object of class 'gapSites'.

Author(s)

Wolfgang Kaisers

Examples

# A) Construct source data from scratch
seqid<-c("chr1","chr1","chr2","chr2","chr2")
lstart<-c(900, 1900,900 ,900, 1900)
lend<-c(1000,2000,1000,1000,2000)
rstart<-c(1100,2100,1100,1200,2100)
rend<-c(1200,2200,1200,1300,2200
nr_aligns<-c(10,20,30,40,10)

# B) Construct gapSites object
gs<-gapSites(seqid,lstart,lend,rstart,rend,nr_aligns=nr_aligns)
gs

# C) Use gapSites accessors
seqid(gs)
lend(gs)
rstart(gs)
strand(gs)
gptm(gs)
rpmg(gs)
nAligns(ga)
nAlignGaps(ga)

# D) Create
bam<-system.file("extdata", "rna_fem.bam", package="spliceSites")
reader<-bamReader(bam,idx=TRUE)
ga<-alignGapList(reader)
ga
dnafile<-system.file("extdata","dna_small.RData",package="spliceSites")
load(dnafile)
dga<-dnaGapSites(ga,dna_small)

gapSites-class

Class "gapSites": Container for tabulated alignment gap positions on RNA-seq data.

Description

Contains tabulated data on alignment gaps on RNA-seq data. "getAlignGaps(reader,seqid)" reads gapped alignments for the specified seqid from a BAM file (via CRAN rbamtools) into an object of class "gapSites".

Objects from the Class

Objects can be created by calls of the form alignGapList(reader).

Slots

nAligns: Object of class "numeric" Total number of aligns in alignment.
nAlignGaps: Object of class "numeric" Total number of gapped aligns in alignment.
dt: Object of class "data.frame" Table containing basic data for object.
annotation: Object of class "dataFrameOrNULL" Optional data.frame containing annotation data.
profile: dataFrameOrNULL Optional. Contains probe information (Name of BAM-file, group affiliation, number of sites).

Methods

as.data.frame signature(x = "gapSites"): Returns copy of contained data.frame.
c signature(x = "gapSites"): Specialisation of generic combine function.
dim signature(x = "gapSites"): Specialisation of generic dim function.
dnaGapSites signature(x = "gapSites", dnaset="DNAStringSet"): Create dnaGapSites object by adding DNA sequences.
getAnnStrand signature(x): Return strand vector based on annotation content.
getProfile signature(x): Return profile table (data.frame) which contains BAM-file names, group affiliation and number of Sites.
gptm signature(x = "gapSites"): Reading accessor for gptm values.
head signature(x = "gapSites"): Specialisation of generic head function.
**lrCodons** signature(x = "gapSites"): Returns gapSites object where lstart and rend positions are truncated toward the next smaller full codon position (used for preparation of translation to amino acid sequence)

**lJunc** signature(x = "gapSites", featlen="numeric", gaplen="numeric", keepStrand="logical") (keepStrand: Values for strand are copied from argument, otherwise all positions are marked as "+". unique: Multiple identical positions (arising from alternative splice sites on the right side) are collapsed to one line (number of sites is counted in "mult"). Position: 0-based position of last exon nucleotide in DNA sequence.

**lrJunc** signature(x = "gapSites", lfeatlen="numeric", rfeatlen="numeric", strand): Returns gapSites object where positions are shifted so that given feature length’s are present for lstart and rend positions (used as preparatory steps for obtaining sensible seq - logo’s on exonic junction regions).

**addGeneAligns** signature(object="gapSites"): Adds number of alignments per gene as new column to alignment gap position table. Annotation tables must be present. Otherwise an error occurs.

**merge** signature(x = "gapSites", y = "ANY"): Specialisation of generic merge (data.frame) function.

**nAligns** signature(object = "gapSites"): Reading accessor for nAligns value.

**nAlignGaps** signature(object = "gapSites"): Reading accessor for nAlignGaps value.

**rpmg** signature(x = "gapSites"): Reading accessor for rpmg values.

**show** signature(object = "gapSites"): Specialisation of generic show function.

**sortTable** signature(x="gapSites"): Sorts all contained tables by seqid, lend and rstart.

**write.annDNA.tables** signature(x="gapSites", dnaset="DNAStringSet", filename="character", featlen="numeric", gaplen="numeric", sep="character", dec="character", row.names="logical") Writes csv file with gap-positions, annotations and dna-sequence.

**Author(s)**

Wolfgang Kaisers

**See Also**

dnaGapSites

**Examples**

```r
bam<-character(2)
bam[1]<-system.file("extdata", "rna_fem.bam", package="spliceSites")
reader<-bamReader(bam[1],idx=TRUE)
agl<-alignGapList(reader)
agl
bamClose(reader)

mbs<-readMergedBamGaps(bam)
mbs
getProfile(mbs)
```
getGapSites  

Description

getGapSites and alignGapList read gap-site data from single BAM-files (given as bamReader) and return a gapSites object. getGapSites reads data for one seqid (given as 1-based numeric value). alignGapList reads the whole BAM-file. The functions test for opened reader and initialized index.

Usage

gapSites(reader, seqid, startid = 1)

Arguments

reader  
.bamReader (rbamtools). An opened instance of bamReader with initialized index.

seqid  
Numeric. 1-based index of reference sequence for which gap-sites are to be read.

startid  
Numeric. Default: 1. Determines start value for id column from which the values are ascending enumerated. startid greater than 1 allow to produce unique values over multiple BAM-files.

Details

gapSites internally calls rbamtools::gapList. alignGapList internally calls rbamtools::bamGapList. 'nProbes' values are set to 1.

Value

gapSites

Author(s)

Wolfgang Kaisers

Examples

bam <- system.file("extdata", "rna_fem.bam", package="spliceSites")
reader <- bamReader(bam, idx=TRUE)
gal <- getGapSites(reader, 1, startid = 10)
gal
gal <- alignGapList(reader)
gal
Description

Provides methods and data for calculation of HBond 5’splice-site scores. HBond scores provide a measure for the capability of a 5’ splice-site to form H-bonds with the U1 snRNA. The function requires at least 3 exon nucleotides and 8 intron nucleotides. The hbond function takes a vector DNA sequences and a vector of position (pos) values. The position values represent the 1-based position of the last exon nucleotide. Therefore all position values must be >= pos+8.

Details

The first two intron nucleotides must be ‘GT’ otherwise returned value is 0. All other sequence characters must be in “ATCG” (capitalization does not matter). When any other character (such as N) is found, the function also returns 0.

Creation of hbond objects

Objects can be created by load.hbond().

Slots

ev: Object of class "environment" Contains external score data.
basedir: Object of class "character" Directory from which external data is restored.

Methods

basedir signature(x = "hbond"): Returns basedir value.
basedir<- signature(x = "hbond",value="character"): Sets basedir value.
hbond signature(x = "hbond",seq="character",pos="integer"): Calculates score5 value for seq at given position.

Author(s)

Wolfgang Kaisers

References

http://www.uni-duesseldorf.de/rna/html/hbond_score.php

Examples

hb<-load.hbond()
seq<-c("CAGGTGAGTTC","ATGCTGGAGAA","AGGGTGCGGGC","AAGGTAACGTC","AAGGTGAGTTC")
hbond(hb, seq, 3)
head-methods  head  Return first lines of contained data.frame.

Description
Methods for function head.

Methods
signature(x = "cRanges")  Method for 'cRanges'.
signature(x = "aaGapSites")  Method for 'aaGapSites'.
signature(x = "cdRanges")  Method for 'cdRanges'.
signature(x = "cRanges")  Method for 'cRanges'.
signature(x = "dnaGapSites")  Method for 'dnaGapSites'.
signature(x = "gapSites")  Method for 'gapSites'.

initialize-methods  initialize  Initializing objects.

Description
Methods for function initialize

Methods
signature(.Object = "cdRanges")  Method for 'cdRanges'.
signature(.Object = "cRanges")  Method for 'cRanges'.
signature(.Object = "keyProfiler")  Method for 'keyProfiler'.
signature(.Object = "SpliceCountSet")  Method for 'SpliceCountSet'.

keyProfiler-class  Class "keyProfiler"

Description
Internal class that counts occurrence of profile factors (e.g. gender male and female) successively for added key-tables. The columns of the key-tables define the groups (e.g. genomic positions: seqid, start, end) for each all profile factors are counted.

Objects from the Class
Objects can be created by calls of the form new("annAligns", ...).
Slots

- `ev`: Environment: contains the main data of each object. The environment contains the data.frames 'dtb' (key-tabled profiles) and 'prof' (profiles: a table that contains the profile definition for each added key-table) as well as 'groupExpr', an unevaluated Expression which does the data.frame-grouping after addition of a new key-table.

- `unique`: Logical: When true, there can be maximal one table added for each indexed profile

- `counted`: Logical: Stores the information which profile already has been counted. Is only used when 'unique' is 'TRUE'.

- `useValues`: Logical: When TRUE, the object tables the values given together with each key-table, otherwise the profiles are simply counted.

Methods

- `addKeyTable` signature(x = "keyProfiler", keyTable="data.frame", index="numeric", values="numeric")
  Adds keyed data to key-table and counts values according to profile (which is defined by index via profile table).

- `getKeyTable` signature(x = "keyProfiler")
  Returns key-table.

- `appendKeyTable` signature(x = "keyProfiler", keytable="data.frame", prefix="character", valFactor="numeric")
  Chinds internal key-table to keytable-argument. A prefix can be added to column-names. A given valFactor causes counted values to be converted into rates (i.e. divided by column-sums and multiplied with rateFactor value. Values are rounded when a digits argument is provided.)

Author(s)

Wolfgang Kaisers

Examples

```r
# Loads profile, position data (key) and aggregated values (ku) data.frames
load(system.file("extdata", "key.RData", package="spliceSites"))
# Group positions
kpc<-new("keyProfiler",keyTable=key1[,c("seqid","lend","rstart")],prof=prof)
addKeyTable(kpc,keyTable=key2[,c("seqid","lend","rstart")],
           index=2,values=key2$nAligns)
addKeyTable(kpc,keyTable=key3[,c("seqid","lend","rstart")],
           index=4,values=key3$nAligns)
cp<-appendKeyTable(kpc,ku,prefix="c."
```
Usage

lrCodons(x, frame=1L, strand="+")

Arguments

x  
gapSites. Object in which codon positions are calculated
frame  
Numeric. Default is 1. Accepted values are 1, 2 or 3. The value causes a frameshift of size (frame-1).
strand  
Character or numeric. Default is '+' which is equivalent to 0. Any other value will be interpreted as '-' which is equivalent to 1.

Details

The function causes an upstream frameshift and a downstream truncation. gapSites objects contain data on gap aligns which represent a related pair of exon-intron boundaries. The returned object is of the same class as the input. Supplemented DNA sequence gapSites objects will omit introns and will represent the 'spliced' DNA around the splice site. lrCodon function is intended to shift coordinates, so that the resulting DNA-sequence can readily be translated in a putative amino-acid sequence which contains the splice-site.

Author(s)

Wolfgang Kaisers

Examples

# A) Create gapSites object
bam<-system.file("extdata","rna_fem.bam",package="spliceSites")
reader<-bamReader(bam,idx=TRUE)
ga<-alignGapList(reader)
bamClose(reader)

# B) lr-Junctions for '+' -strand
lrj<-lrJunc(ga,lfeatlen=6,rfeatlen=6,strand='+')
lr1<-lrCodons(lrj,frame=1)
lr2<-lrCodons(lrj,frame=2)
lr3<-lrCodons(lrj,frame=3)
lr<-c(lr1,lr2,lr3)

# C) lr-Junctions for '-' -strand
lrj<-lrJunc(ga,lfeatlen=6,rfeatlen=6,strand='-')
lr1<-lrCodons(lrj,frame=1)
lr2<-lrCodons(lrj,frame=2)
lr3<-lrCodons(lrj,frame=3)
lr<-c(lr1,lr2,lr3)
maxEnt-class

Class "maxEnt"

Description

Provides methods for calculation of Splice-site scores. Both functions (score5 and score3) are intended to work on the '+' strand. score5 scores the 5' side (i.e. the splice donor, left) and the score3 scores the 3' side (i.e. the splice acceptor, right).

Creation of maxEnt objects

Objects can be created by load.maxEnt().

Slots

ev: Object of class "environment" Contains external score data.

basedir: Object of class "character" Directory from which external data is restored.

Methods

basedir signature(x = "maxEnt"): Returns basedir value.

basedir<- signature(x = "maxEnt", value = "character"): Sets basedir value.

score5 signature(x = "maxEnt", seq = "character", pos = "integer"): Calculates score5 value for seq at given position.

scoreSeq5 signature(x = "maxEnt", seq = "character", frame = "integer"): Calculates score5 values for a single sequence and a series of positions (frame).

score3 signature(x = "maxEnt", seq = "character", pos = "integer", which = "character"): Calculates score3 value for seq at given position. Accepted values for which are: "ent", "wmm" and "emm".

scoreSeq3 signature(x = "maxEnt", seq = "character", frame = "integer", which = "character"): Calculates score3 values for a single sequence and a series of positions (frame). Accepted values for which are: "ent", "wmm" and "emm".

Author(s)

Wolfgang Kaisers

Examples

mes<-load.maxEnt()
score5(mes,"CCGGTAAAGA",4) # 9.844127
score3(mes,"CTCTACTATCTATCTAGATC",pos=20) # 6.706947

# scoreSeq functions
sq5<-scoreSeq5(mes,seq="ACGGTAAGTCAGGTAAGT")
sq3<-scoreSeq3(mes,seq="TTTATTTTTCTCAGTTAGACTTCTTTCTCATAAGGTT")
merge-methods

merge Merging two objects into one.

Description
Methods for function merge

Methods
signature(x = "gapSites", y = "ANY") Method for 'gapSites'.

plotGeneAlignDepth

plotGeneAlignDepth: Plots of read alignment depth for genetic regions

Description
The function takes a bamReader and a refGenome object together with a gene name and an optional transcript name and plots the read alignment depth for the region of the gene in the opened BAM file. When transcript data is present, the exonic ranges are added as rectangles on a chromosomal line.

Usage
plotGeneAlignDepth(reader, genome, gene=NULL, transcript=NULL, log="y", cex.main=2, col="grey50", fill="grey90", grid=TRUE, box.col="grey20", box.border="grey80")

Arguments
reader bamReader (rbamtools). Must be opened and have initialized index.
genome refGenome. Object which contains genomic annotation data.
gene character. Name of one single gene.
transcript character (optional). Name of one single transcript.
log character. Name of one single gene.
cex.main numeric. Determines size of main title.
col color. A color for align depth line.
fill color. A color for the interior of align depth area.
grid logical. When TRUE, a grid is drawn.
box.col color. A color for the interior of exon rectangles.
box.border color. A color for the border of exon rectangles.

Details
The function checks for opened bamReader and initialized index. When transcript name is given, the function will plot the positions of the transcript beneath the alignment depth.
rangeByGeneName

Author(s)
Wolfgang Kaisers

Examples

# Open bamReader
bam <- system.file("extdata","rna_fem.bam",package="spliceSites")
reader <- bamReader(bam, idx=TRUE)
# Load annotation data
ucf <- system.file("extdata", "uc_small.RData", package="spliceSites")
uc <- loadGenome(ucf)
plotGeneAlignDepth(reader, uc, gene="WASH7P", transcript="uc001aac.4")

rangeByGeneName

Reads a bamRange object for a given bamReader, refGenome and gene name.

Description
Locates gene in genome via refGenome and reads a bamRange from the determined region.

Usage
rangeByGeneName(reader, genome, gene, complex=TRUE)

Arguments
reader Object of class (rbamtools) bamReader. The reader must pass isOpen and
index.initialized test.
genome Object of class (refgenome) refGenome.
gene Single gene name (character)
complex Logical. Passed to 'bamRange' function. When TRUE, only aligns with nCigar>1 are counted.

Value
bamRange

Author(s)
Wolfgang Kaisers

Examples

bam<-system.file("extdata","rna_fem.bam",package="spliceSites")
reader<-bamReader(bam,idx=TRUE)
ucf<-system.file("extdata","uc_small.RData",package="spliceSites")
uc<-loadGenome(ucf)
range<-rangeByGeneName(reader,uc,"WASH7P")
size(range)
Description

Opens fpkm_tracking files and collects FPKM values into ExpressionSet. The function is intended to work with genes.fpkm_tracking files. In order to get unique gene identifier, the contained values are grouped and for each gene the maximum FPKM values is selected. There should only be a few hundred multiple occurring genes and the maximum value should give a (slight) underestimation of the real value.

Usage

readCuffGeneFpkm(cuff, phenoData, summ="max")

Arguments

cuff character: Vector of cufflinks files
phenoData AnnotatedDataFrame: Requirement for construction of an ExpressionSet
summ character: Must be either 'max' or 'sum'. A handful of tracking id's occur multiple times due to multiple transcripts which partially are non-overlapping. The summ (summarize) Argument determines the way the multiplets are handled.

Value

ExpressionSet

Author(s)

Wolfgang Kaisers

Examples

n<-10
cuff <- system.file("extdata", "cuff_files", paste(1:n, "genes", "fpkm_tracking", sep="."), package="spliceSites")

# Create Pheno - data
gr <- system.file("extdata", "cuff_files", "groups.csv", package="spliceSites")
groups <- read.table(gr, sep="\t", header=TRUE)
meta <- data.frame(labelDescription=c("gender", "age-group", "location"),
row.names=c("gen", "agg", "loc"))
phenoData <- new("AnnotatedDataFrame", data=groups, varMetadata=meta)

# Read ExpressionSet
exset <- readCuffGeneFpkm(cuff, phenoData)
**readExpSet**  
Reads align number or gptm or rpmg value from all given BAM-files and all identified align gaps into ExpressionSet.

### Description
Opens multiple BAM-files and reads aligns for selected gene for each file. Number of alignes is counted.

### Usage

```
readExpSet(bam, idx, val="nAligns", phenoData, expData)
```

### Arguments
- **bam**: Vector of BAM-files
- **idx**: Vector of index files (optional)
- **val**: "gptm", "rpmg" or "nAligns". Value type which is written to ExpressionSet matrix (nAligns = read count).
- **phenoData**: AnnotatedDataFrame. Each BAM-file must correspond to one identifier.
- **expData**: MIAME. Optional. Experiment data which can be added to ExpressionSet

### Value
ExpressionSet

### Author(s)
Wolfgang Kaisers

### Examples

```
# A) Names of BAM-files
bam <- character(2)
bam[1] <- system.file("extdata", "rna_fem.bam", package="spliceSites")

# B) Experiment Profile
prof <- data.frame(gender=c("f", "m"))
meta <- data.frame(labelDescription=names(prof), row.names=names(prof))
pd <- new("AnnotatedDataFrame", data=prof, varMetadata=meta)

# C) Read ExpressionSet
es <- readExpSet(bam, phenoData=pd)

# D) Annotate ExpressionSet
ucf <- system.file("extdata", "uc_small_junc.RData", package="spliceSites")
juc <- loadGenome(ucf)
ann <- annotate(es, juc)
phenoData(es) <- ann
```
### readMergedBamGaps

**Description**

The function takes a vector of BAM-file names and corresponding BAM-index file names. For each given filename, the BAM-file will be opened. The function uses the `bamGapList` function (rbamtools) to obtain a data.frame from `bamReader`. Values for 'gptm' and 'rpmg' are added. Both are rounded to the number of given digits. The function tests for open connection to BAM-file and for initialized index.

**Usage**

```r
readMergedBamGaps(infiles, idxInfiles=paste(infiles,".bai",sep=""), digits=3)
```

**Arguments**

- `infiles` character. Name of BAM-files to be opened.
- `idxInfiles` character. Name of corresponding BAM-index files. Default: `paste(infiles,".bai",sep="")`
- `digits` numeric. `gptm` and `rpmg` values will be rounded to the number of decimal places given.

**Value**

gapSites

**Author(s)**

Wolfgang Kaisers

**Examples**

```r
bam<-character(2)
bam[1]<-system.file("extdata","rna_fem.bam",package="spliceSites")
bam[2]<-system.file("extdata","rna_mal.bam",package="spliceSites")
mbg<-readMergedBamGaps(bam)
```

---

### readTabledBamGaps

**Description**

readTabledBamGaps

**Usage**

```r
readTabledBamGaps(infiles, idxInfiles=paste(infiles,".bai",sep=""), prof, rpmg=TRUE)
```
Arguments

infiles character. Names of BAM-files.
idxInfiles character. Names of BAM-index files. When given index file is not found, the function attempts to create a BAM-index file with the depicted name.
prof data.frame. Contains group affiliations for each BAM-file. Each column describes an entity by which values are grouped. The row-number in prof must be equal to the number of given BAM-files. The order of BAM infiles and prof defines the group classification for each BAM file. All prof columns must be factors.
rpmg logical. When TRUE, there will be group specific rpmg align-rates be added to the result table

Details

The function reads gap-align data from all given BAM-files. For each factor level, the number of probes and aligns are counted. When gptm=TRUE also the gptm values are written for each group. The result table contains for each prof factor level 2 (or 3) extra columns.

Value

gapSites

Author(s)

Wolfgang Kaisers

Examples

bam<-character(2)
bam[1]<-system.file("extdata","rna_fem.bam",package="spliceSites")
bam[2]<-system.file("extdata","rna_mal.bam",package="spliceSites")
prof<-data.frame(gender=c("f","m"))
rtbg<-readTabledBamGaps(bam,prof=prof,rpmg=TRUE)
rtbg
getProfile(rtbg)

seqlogo

seqlogo: Plotting sequence logo for 'cdRanges' and 'dnaGapSites' objects.

Description

The function produces a sequence logo plot based on the contained sequences.

Usage

seqlogo(x,strand="+",useStrand=TRUE,...)
Arguments

- `x` cdRanges or dnaGapSites Object.
- `strand` Character. Determines the subset for which the seqlogo is plotted. This option is only used when useStrand is given as 'TRUE'.
- `useStrand` Logical. Determines whether the given strand information is used. For `useStrand=FALSE` the plot is made up from all contained sequences.
- `...` (Currently unused)

Details

The function fails with an error message when the dataset does not contain any records with the given strand (except `useStrand=FALSE`).

Value

None

Author(s)

Wolfgang Kaisers

Examples

```r
bam <- system.file("extdata","rna_fem.bam",package="spliceSites")
reader <- bamReader(bam, idx=TRUE)
ga <- alignGapList(reader)
bamClose(reader)
lj <- lJunc(ga, featlen=6, gaplen=6, strand='+) 
dnafile <- system.file("extdata","dna_small.RData",package="spliceSites")
load(dnafile)
ljd <- dnaRanges(lj, dna_small)
seqlogo(ljd)
```

---

**silic_tryp**

**silic_tryp function**

Description

`silic_tryp` performs silicon trypsination and returns the fragments to which the position coordinate points. The position value is corrected so that it afterwards points to the same amino-acid as before.

Usage

`silic_tryp(seq, pos, id)`

Arguments

- `seq` Character. Amino-acid sequences which are to be truncated.
- `pos` Numeric. Points to an amino-acid inside the sequence.
- `id` Numeric. An identifier which is copied to the result table.
sortTable-methods

Details

The routine implements the "Keil"-rule, where sites are described by the regex "([RK](?!P))". The cut position is between [RK] and the following character. The sequence fragment which contains the exon-intron boundary (depicted by position) is returned. Dependent numeric values are recalculated.

Value
data.frame

Author(s)
Wolfgang Kaisers

Examples

silic_tryp(seq="AXKUEMRFG",pos=4)

---

Description

Sorting contained data with sortTable.

Methods

signature(x = "cRanges") Method for 'cRanges'. Key columns: seqid, start, end
signature(x = "gapSites") Method for 'gapSites'. Key columns: seqid, lend, rstart

---

SpliceCountSet-class

Class "SpliceCountSet"

Description

Directly inherits from ExpressionSet

Objects from the Class

Objects can be created by calls of the form new("SpliceCountSet", ...).

Author(s)
Wolfgang Kaisers

Examples

# scs<-new("SpliceCountSet")
trim and resize methods: trim_left, trim_right, resize_left, resize_right

### Description

The trim and resize functions change number of nucleotides contained in align-gap features (exonic). Trim functions cut feature sizes down to `maxlen`. Resize functions reset all sizes to a fixed value. The functions operate directly on the passed objects. There is no return value.

### Usage

`trim_left(x, maxlen)`

### Arguments

- `x`: gapSites. Object from which the IUnc values are calculated.
- `maxlen`: Numeric. Maximum number of nucleotides on feature (exon) side of boundary.

### Value

None.

### Author(s)

Wolfgang Kaisers

### Examples

```r
# A) Create gapSites object
bam <- system.file("extdata", "rna_fem.bam", package="spliceSites")
reader <- bamReader(bam[1], idx=TRUE)
ga <- alignGapList(reader)
bamClose(reader)
ga

# B) Trim
trim_left(ga, 3)
trim_right(ga, 2)
ga

# C) Resize
resize_left(ga, 5)
resize_right(ga, 6)
ga
```
### truncateSeq

**Description**

truncateSeqs amino acid sequences at positions depicted by '*' (stop-codon).

**Usage**

```r
truncateSeq(x, rme=TRUE, trunc=42L)
```

**Arguments**

- `x`: caRanges. Object in which amino-acid sequences are to be truncated.
- `rme`: Logical. Default is TRUE. When TRUE, sites with resulting empty sequence (i.e. stop-codon upstream of the splice position) are removed from dataset.
- `trunc`: Integer. ASCII code for character at which truncation should occur. Default value is 42='*' (stop-codon).

**Details**

The function `truncateSeqs` the contained amino acid sequences. When the stop-codon is found on the left side of position, the function returns an empty sequence for that site. The position values for these records are also set to 0.

**Value**

Object of same class as input.

**Author(s)**

Wolfgang Kaisers

**Examples**

```r
# A) Read gap-sites from BAM-file
bam<-system.file("extdata","rna_fem.bam",package="spliceSites")
reader<-bamReader(bam,idx=TRUE)
ga<-alignGapList(reader)
bamClose(reader)
# B) Load DNA sequence
dnafile<-system.file("extdata","dna_small.RData",package="spliceSites")
load(dnafile)
ucf<-system.file("extdata","uc_small.RData",package="spliceSites")
uc<-loadGenome(ucf)
# C) Calculate codon frame data and add DNA
lj<-ljunc(ga,featlen=21,gaplen=21,strand='+')
ljc<-lCodons(lj,frame=1,keepStrand=TRUE)
cdr<-dnaRanges(ljc,dna_small)
# D) Translate DNA to amino acid and truncate
ar<-translate(cdr)
tra<-truncateSeq(ar)
```
**Description**

`truncateSeqs` amino acid sequences at positions depicted by `'*'` (stop-codon).

**Usage**

`truncate_seq(seq,pos,id,rme=TRUE,trunc=42L)`

**Arguments**

- `seq`: Character. Amino-acid sequences which are to be truncated.
- `pos`: Numeric. Points to an amino-acid inside the sequence.
- `id`: Numeric. An identifier which is copied to the result table.
- `rme`: Logical. Empty sequences are removed when set to TRUE.
- `trunc`: Integer. ASCII code for character at which truncation should occur. Default value is 42=`'*'` (stop-codon).

**Details**

The function `truncateSeqs` the contained amino acid sequences. When the stop-codon is found on the left side of position, the function returns an empty sequence for that site. The position values for these records are also set to 0.

**Value**

`data.frame`

**Author(s)**

Wolfgang Kaisers

**Examples**

`truncate_seq(seq="ARPX*QR",pos=3)`
Description

trypsinCleave amino acid sequences and returns the fragment which contains the position described by position entry in data.frame.

Usage

trypsinCleave(x, minLen=5, ...)

Arguments

- **x**: caRanges (aaGapSites). Object in which amino-acid sequences are to be truncated.
- **minLen**: Numeric. Default is 5. Data sets where the remaining sequence fragment is shorter than \(\text{minLen}\) are excluded.
- **...**: Additional arguments which may be passed to the routine (currently unused).

Details

The routine implements the "Keil"-rule, where sites are described by the regex "[RK](?!P)". The cut position is between [RK] and the following character. The sequence fragment which contains the exon-intron boundary (depicted by position) is returned. Dependent numeric values are recalculated. The returned sequence ends on "[RK]" unless the returned fragment is a sequence suffix.

Value

Same class as given object.

Author(s)

Wolfgang Kaisers

Examples

```r
bam<-system.file("extdata","rna_fem.bam",package="spliceSites")
reader<-bamReader(bam,idx=TRUE)
ga<-alignGapList(reader)
bamClose(reader)

lj<-lJunc(ga,featlen=21,gaplen=21,strand=quotesingle.Var+quotesingle.Var)
ljc<-lCodons(lj,frame=1,keepStrand=TRUE)
dnafile<-system.file("extdata","dna_small.RData",package="spliceSites")
load(dnafile)
cdr<-dnaRanges(ljc,dna_small)
ar<-translate(cdr)
tra<-truncateSeq(ar)
tycc<-trypsinCleave(tra)
```
Description

uniqueJuncAnn adds annotation data to ExpressionSet and removes not-matching sites.

Usage

uniqueJuncAnn(object, junc, ann = TRUE, ...)

Arguments

- **object**: ExpressionSet. Object containing gap-site expression data.
- **junc**: refJunctions. Object containing splice-junction sites.
- **ann**: logical. Default: TRUE. When TRUE the unannotated sites are removed, otherwise the annotated sites are removed.
- **...**: Unused.

Value

ExpressionSet

Author(s)

Wolfgang Kaisers

Examples

# A) Names of BAM-files
bam <- character(2)
bam[1] <- system.file("extdata","rna_fem.bam",package="spliceSites")
bam[2] <- system.file("extdata","rna_mal.bam",package="spliceSites")

# B) Experiment Profile
prof <- data.frame(gender = c("f","m"))
meta <- data.frame(labelDescription = names(prof), row.names = names(prof))
pd <- new("AnnotatedDataFrame", data = prof, varMetadata = meta)

# C) Read ExpressionSet
es <- readExpSet(bam, phenoData = pd)

# D) Annotate ExpressionSet
ucf <- system.file("extdata","uc_small.RData",package="spliceSites")
uc <- loadGenome(ucf)
ucj <- getSpliceTable(uc)

# E) Extract unique annotated junction sites.
uja <- uniqueJuncAnn(es, ucj)
write.files

**Description**

Writes table data and sequence in separate files.

**Usage**

```
write.files(x, path, filename,...)
```

**Arguments**

- `x`: caRanges or aaGapSites object for which data is written.
- `path`: Path for writing files.
- `filename`: Basic filename to which suffixes are added.
- `...`: Other arguments passed to "write.table".

**Details**

There are two files written: A text file with tabulated values from data.frame (separated by ";") and a fasta file which contains the stored dna sequence.

**Value**

None.

**Note**

The function tries to create directory 'path' when it does not exist.

**Author(s)**

Wolfgang Kaisers

**Examples**

```
# A) Read gap-sites from BAM-files
bam<-'system.file("extdata","rna_fem.bam",package="spliceSites")
reader<-'bamReader(bam,idx=TRUE)
ga<-'alignGapList(reader)
bamClose(reader)
# B) Load DNA sequence
dnafile<-'system.file("extdata","dna_small.RData",package="spliceSites")
load(dnafile)
# C) Add DNA sequence
lj<-lJunc(ga,featlen=21,gaplen=21,strand='+')
ljc<-lCodons(lj,frame=1,keepStrand=TRUE)
cdr<-'dnaRanges(ljc,dna_small)
# D) Translate DNA to amino-acid
ar<-'translate(cdr)
# E) Write "ar.csv" and "ar.fa"
write.files(ar,"","ar")
```
xCodons

### Description

The xCodon functions work on cRanges objects. On the contained ranges the function can have two effects: an upstream frame-shift of 0 to 2 positions and a downstream trim to full codons (i.e. \((\text{end}-\text{start}+1)\%3==0\)). The lCodon function acts in `+' strand mode (left frame-shift, right truncation) and the rCodon function acts in `-' strand mode (right frame-shift, left truncation).

### Usage

```r
1Codons(x, frame=1, keepStrand=TRUE)
```

### Arguments

- **x**: cRanges. Object in which codon positions are calculated
- **frame**: Numeric. Default is 1. Accepted values are 1, 2 or 3. The value causes a frame-shift of size \((\text{frame}-1)\).
- **keepStrand**: Logical. Default is TRUE. When FALSE, lCodons overwrites strand entries by '+' and rCodons overwrites strand entries by '-'.

### Details

The function causes an upstream framesshift and a downstream truncation. lCodon works with `+'-strand view (left-to-right) and rCodon works with `-'-strand view (right to left). The underlying rationale is: The cRanges object contains ranges around exon-intron boundaries. The boundary itself is marked by the position value. The functions calculate genomic ranges which can be supplemented by the reference DNA-sequence which then can readily be translated into amino-acid sequences. The different values for frame and keepStrand are used to produce all six putative amino-acid sequences for this exon-intron boundary.

### Author(s)

Wolfgang Kaisers

### Examples

```r
bam<-system.file("extdata","rna_fem.bam",package="spliceSites")
reader<-bamReader(bam,idx=TRUE)
ja<-alignGapList(reader)
bamClose(reader)
dnafile<-system.file("extdata","dna_small.RData",package="spliceSites")
load(dnafile)
ucf<-system.file("extdata","uc_small.RData",package="spliceSites")
uc<-loadGenome(ucf)

lj<-lJunc(ja,featlen=21,gaplen=21,strand=\"+/\"
 lj<-1Codons(lj,frame=1,keepStrand=TRUE)
 rj<-rJunc(ja,featlen=21,gaplen=21,strand=\"-/\"
 rj<-rCodons(rj,frame=1,keepStrand=TRUE)```
xJunc

xJunc methods: lJunc, rJunc, lrJunc

Description

The term 'xJunc' envelopes three functions: lJunc, rJunc and lrJunc. All three functions take a gapSites object and return ranges which are restricted around align-gap (exon-intron) boundaries. The functions lJunc and rJunc return cRanges objects, the lrJunc function returns a gapSites object.

Usage

lJunc(x, featlen, gaplen, unique=FALSE, strand, ...)

Arguments

x         gapSites. Object from which the lJunc values are calculated.
featlen   Numeric. Number of nucleotides on feature (exon) side of boundary.
gaplen    Numeric. Number of nucleotides on gap (intron) side of boundary.
unique    Logical. Default is 'FALSE'. When 'TRUE', the function removes duplicate entries which can be due to alternative splice events.
strand    Character. Mandatory. All strand entries are set to the given value.
...       Optional arguments passed additionally to the function (currently unused).

Details

The functions are intended to provide position information which crosses exon-intron boundaries. Added DNA sequences can be used to produce seqlogos. The functions are intended to be used in advance of xCodons functions. Later on added AA sequences can be used to search for proteins where intronic sequences are retained.

Value

cRanges

Author(s)

Wolfgang Kaisers

Examples

# A) Create gapSites object
bam<-'system.file("extdata","rna_fem.bam",package="spliceSites")
reader<-'bamReader(bam[1],idx=TRUE)
ga<-alignGapList(reader)
bamClose(reader)
ga

# B) Extract junction data
lj<-lJunc(ga, featlen=6, gaplen=6, strand='+')
ljm<-lJunc(ga, featlen=6, gaplen=6, strand='-')
rj < - rJunc(ga, featlen = 6, gaplen = 6, strand = '+')
rm < - rJunc(ga, featlen = 6, gaplen = 6, strand = '-')
lrj < - lrJunc(ga, lfeatlen = 6, rfeatlen = 6, strand = '+')
lrm < - lrJunc(ga, lfeatlen = 6, rfeatlen = 6, strand = '-')

xJuncStrand methods: lJuncStrand, rJuncStrand, lrJuncStrand

Description

The term 'xJuncStrand' envelopes three functions: lJuncStrand, rJuncStrand and lrJuncStrand. All three functions take a gapSites object and return ranges which are restricted around align-gap (exon-intron) boundaries. The functions lJuncStrand and rJuncStrand return cRanges objects, the lrJuncStrand function returns a gapSites object. The resulting objects contain strand information which is copied from the input objects.

Usage

lJuncStrand(x, featlen, gaplen, ...)

Arguments

x  gapSites. Object from which the lJuncStrand values are calculated.
featlen  Numeric. Number of nucleotides on feature (exon) side of boundary.
gaplen  Numeric. Number of nucleotides on gap (intron) side of boundary.
...  Optional arguments passed additionally to the function (currently unused).

Details

The functions are intended to provide position information which crosses exon-intron boundaries. Added DNA sequences can be used to produce seqlogos. The functions are intended to be used in advance of xCodons functions. Later on added AA sequences can be used to search for proteins where intronic sequences are retained.

Value

cRanges

Author(s)

Wolfgang Kaisers

Examples

# A) Create gapSites object
bam <- system.file("extdata","rna_fem.bam",package="spliceSites")
reader <- bamReader(bam[1], idx = TRUE)
ga <- alignGapList(reader)
bamClose(reader)

# B) Extract JuncStrandtion data
lj<-lJuncStrand(ga,featlen=6,gaplen=6)
ljm<-lJuncStrand(ga,featlen=6,gaplen=6)
rj<-rJuncStrand(ga,featlen=6,gaplen=6)
rjm<-rJuncStrand(ga,featlen=6,gaplen=6)
lrj<-lrJuncStrand(ga,lfeatlen=6,rfeatlen=6)
lrjm<-lrJuncStrand(ga,lfeatlen=6,rfeatlen=6)

[-methods

Methods for Function '['

Description

Methods for function ['

Methods

signature(x = "cRanges",i="ANY",j="ANY",drop="ANY") Method for 'cRanges'.
signature(x = "gapSites",i="ANY",j="ANY",drop="ANY") Method for 'gapSites'.


Index

*Topic **ExpressionSet**
  annotate-ExpressionSet, 10
  readCuffGeneFpkm, 34
  readExpSet, 35
*Topic **SpliceCountSet**
  SpliceCountSet-class, 39
*Topic **addGeneAligns**
  addGeneAligns, 5
*Topic **addMaxEnt**
  addMaxEnt, 7
*Topic **alignGapList**
  getGapSites, 26
*Topic **annGapSites**
  annGapSites-class, 9
*Topic **annotate**
  annotate-ExpressionSet, 10
  annotation, 11
*Topic **annotation**
  annotation, 11
*Topic **as.data.frame**
  as.data.frame-methods, 12
*Topic **bamCount**
  countByGeneName, 15
*Topic **bamRange**
  rangeByGeneName, 33
*Topic **bamReader**
  getGapSites, 26
*Topic **cRanges**
  cRanges-class, 16
  extractByGeneName, 20
  extractRange, 21
*Topic **caRanges**
  caRanges-class, 13
  extractByGeneName, 20
  extractRange, 21
*Topic **cdRanges**
  cdRanges-class, 14
  extractByGeneName, 20
  extractRange, 21
*Topic **classes**
  aaGapSites-class, 3
  caRanges-class, 13
  cdRanges-class, 14
  cRanges-class, 16
  dnaGapSites-class, 18
  gapSites-class, 24
  hbond-class, 27
  maxEnt-class, 31
*Topic **c**
  c-methods, 12
*Topic **dim**
  dim-methods, 18
*Topic **dnaRanges**
  dnaRanges, 19
*Topic **extractByGeneName**
  extractByGeneName, 20
*Topic **extractRange**
  extractRange, 21
*Topic **gapAligns**
  addGeneAligns, 5
*Topic **gapSites**
  alt_X_ranks, 8
  annotation, 11
  countByGeneName, 15
  extractByGeneName, 20
  extractRange, 21
  gapSites, 22
  gapSites-class, 24
  getGapSites, 26
  readMergedBamGaps, 36
*Topic **hbond**
  addHbond, 6
*Topic **head**
  head-methods, 28
*Topic **initialize**
  initialize-methods, 28
*Topic **keyProfiler**
  keyProfiler-class, 28
  readTabledBamGaps, 36
*Topic **lCodons**
  trim, 40
  xCodons, 46
  xJunc, 47
  xJuncStrand, 48
*Topic **lrCodons**
  lrCodons, 29
*Topic `maxEnt`
  addMaxEnt, 7

*Topic `merge`
  merge-methods, 32

*Topic `methods`
  [-methods, 49
  as.data.frame-methods, 12
  c-methods, 12
  dim-methods, 18
  head-methods, 28
  initialize-methods, 28
  merge-methods, 32
  sortTable-methods, 39

*Topic `package`
  spliceSites-package, 2

*Topic `plotGeneAlignDepth`
  plotGeneAlignDepth, 32

*Topic `rCodons`
  trim, 40
  xCodons, 46
  xJunc, 47
  xJuncStrand, 48

*Topic `rangeByGeneName`
  rangeByGeneName, 33

*Topic `readCuffGeneFpkm`
  readCuffGeneFpkm, 34

*Topic `readExpSet`
  readExpSet, 35

*Topic `readMergedBamGaps`
  readMergedBamGaps, 36

*Topic `readTabledBamGaps`
  readTabledBamGaps, 36

*Topic `refGenome`
  annotation, 11
  rangeByGeneName, 33

*Topic `seqlogo`
  seqlogo, 37

*Topic `silic_tryp`
  silic_tryp, 38

*Topic `sortTables`
  sortTable-methods, 39

*Topic `stop-codon`
  trim, 40
  truncateSeq, 41
  uniqueJuncAnn, 44
  xCodons, 46
  xJunc, 47
  xJuncStrand, 48

*Topic `truncateSeq`
  truncateSeq, 41

*Topic `truncate_seq`
  truncate_seq, 42

*Topic `trypsinCleave`
  trypsinCleave, 43

*Topic `uniqueJuncAnn`
  uniqueJuncAnn, 44

*Topic `write.files`
  write.files, 45
  [-methods, 49
  aaGapSites-class, 3
  addGeneAligns, 5
  addGeneAligns,gapSites-method (addGeneAligns), 5
  addHbond, 6
  addHbond,cdRanges-method (addHbond), 6
  addHbond,gapSites-method (addHbond), 6
  addHbond-methods (addHbond), 6
  addKeyTable (keyProfiler-class), 28
  addKeyTable,keyProfiler-method (keyProfiler-class), 28
  addKeyTable-methods (keyProfiler-class), 28
  addMaxEnt, 7
  addMaxEnt,gapSites-method (addMaxEnt), 7
  addMaxEnt-methods (addMaxEnt), 7
  alignGapList (getGapSites), 26
  alt_left_ranks (alt_X_ranks), 8
  alt_left_ranks,gapSites-method (alt_X_ranks), 8
  alt_left_ranks-methods (alt_X_ranks), 8
  alt_ranks (alt_X_ranks), 8
  alt_ranks,gapSites-method (alt_X_ranks), 8
  alt_ranks-methods (alt_X_ranks), 8
  alt_right_ranks (alt_X_ranks), 8
  alt_right_ranks,gapSites-method (alt_X_ranks), 8
  alt_right_ranks-method (alt_X_ranks), 8
  alt_X_ranks, 8
  annGapSites-class, 9
  annotate (annotation), 11
  annotate,ExpressionSet,refJunctions-method (annotate-ExpressionSet), 10
  annotate,gapSites,refJunctions-method (annotation), 11
  annotate-ExpressionSet, 10
  annotate-methods (annotation), 11
  annotation, 11
  annotation,gapSites-method (annotation), 11
  annotation-methods (annotation), 11
  annotation<- (annotation), 11
INDEX

annotation<-,gapSites,ANY-method (annotation), 11
annotation<-,gapSites,refJunctions-method (annotation), 11
annotation<-,methods (annotation), 11
appendKeyTable (keyProfiler-class), 28
appendKeyTable (keyProfiler-class), 28
appendKeyTable-methods (keyProfiler-class), 28
appendKeyTable-methods (keyProfiler-class), 28
as.data.frame.gapSites (as.data.frame-methods), 12
as.data.frame.cRanges (cRanges-class), 16
as.data.frame.gapSites (gapSites-class), 24
basedir, hbond-method (hbond-class), 27
basedir, maxEnt-method (maxEnt-class), 31
basedir<-,hbond-method (hbond-class), 27
basedir<-,maxEnt-method (maxEnt-class), 31
c, caRanges-method (caRanges-class), 13
c, cdRanges-method (cdRanges-class), 14
c, cRanges-method (cRanges-class), 16
c, gapSites-method (gapSites-class), 24
c-methods, 12
cRanges-class, 13
cdRanges-class, 14
count (cRanges-class), 16
count, cRanges-method (cRanges-class), 16
count-methods (cRanges-class), 16
countByGeneName, 15
cRanges, 13, 14
cRanges-class, 16
dim, cRanges-method (cRanges-class), 16
dim, gapSites-method (gapSites-class), 24
dim-methods, 18
dnaGapSites (dnaGapSites-class), 22
dnaGapSites, gapSites, DNAStringSet-method (gapSites-class), 24
dnaGapSites-class, 18
dnaGapSites-methods (gapSites-class), 22
dnaRanges, 19
dnaRanges, gapSites, DNAStringSet, logical-methods (dnaRanges), 19
dnaRanges, cRanges, DNAStringSet, missing-method (dnaRanges), 19
dnaRanges-methods (dnaRanges), 19
do_group_align_data (gapSites-class), 24
dim-methods (cRanges-class), 16
extractByGeneName, 20
eextractByGeneName, cRanges, ANY-method (extractByGeneName), 20
eextractByGeneName, cRanges-method (extractByGeneName), 20
eextractByGeneName, gapSites, ANY-method (extractByGeneName), 20
eextractByGeneName, gapSites-method (extractByGeneName), 20
extractByGeneName-methods (extractByGeneName), 20
extractRange, 21
extractRange, cRanges-method (extractRange), 21
extractRange, gapSites-method (extractRange), 21
extractRange-methods (extractRange), 21
gapSites, 4, 9, 18, 22
gapSites-class, 24
gapSites-class
gapSites-class
gapSites-class
getAnnStrand (gapSites-class), 24
getAnnStrand, gapSites-method (gapSites-class), 24
getAnnStrand-methods (gapSites-class), 24
getGAPsites, 26
getKeyTable (keyProfiler-class), 28
g getKeyTable, keyProfiler-method (keyProfiler-class), 28
g getKeyTable-methods (keyProfiler-class), 28
getMeStrand (gapSites-class), 7
getMeStrand, gapSites-method (addMaxEnt), 7
getMeStrand-methods (addMaxEnt), 7
g PROFILE (gapSites-class), 24
g PROFILE, gapSites-method (gapSites-class), 24
g PROFILE-methods (gapSites-class), 24
gSequence (cdRanges-class), 14
g getSequence, caRanges-method (caRanges-class), 13
g getSequence, cdRanges-method (cdRanges-class), 14
g getSequence-methods (cdRanges-class), 14
g ptm (gapSites-class), 24
gptm, gapSites-method (gapSites-class), 24
g ptm-methods (gapSites-class), 24
hbond (hbond-class), 27
hbond, hbond-method (hbond-class), 27
hbond-class, 27
hbond-methods (hbond-class), 27
head,aaGapSites-method
  (aaGapSites-class), 3
head,caRanges-method (caRanges-class), 13
head,cdRanges-method (cdRanges-class), 14
head,dnaGapSites-method
  (dnaGapSites-class), 18
head,gapSites-method
  (gapSites-class), 24
head-methods, 28
id (cRanges-class), 16
id,cRanges-method (cRanges-class), 16
initialize,cdRanges-method
  (cdRanges-class), 14
initialize,cRanges-method
  (cRanges-class), 16
initialize,keyProfiler-method
  (keyProfiler-class), 28
initialize,SpliceCountSet-method
  (SpliceCountSet-class), 39
initialize-methods, 28
keyProfiler-class, 28
lCodons (xCodons), 46
lCodons,cRanges-method (xCodons), 46
lCodons-methods (xCodons), 46
lend (gapSites-class), 24
lend,gapSites-method (gapSites-class), 24
lend-methods (gapSites-class), 24
lJunc (xJunc), 47
lJunc,gapSites-method (xJunc), 47
lJunc-methods (xJunc), 47
lJuncStrand (xJuncStrand), 48
lJuncStrand,gapSites-method
  (xJuncStrand), 48
lJuncStrand-methods (xJuncStrand), 48
load.hbond (hbond-class), 27
load.maxEnt (maxEnt-class), 31
lrCodons, 29
lrCodons,gapSites-method (lrCodons), 29
lrCodons-methods (lrCodons), 29
lrJunc (xJunc), 47
lrJunc,gapSites-method (xJunc), 47
lrJunc-methods (xJunc), 47
lrJuncStrand (xJuncStrand), 48
lrJuncStrand,gapSites-method
  (xJuncStrand), 48
lrJuncStrand-methods (xJuncStrand), 48
maxEnt-class, 31
merge-methods, 32
merge.gapSites (gapSites-class), 24
nAlignGaps (gapSites-class), 24
nAlignGaps,gapSites-method
  (gapSites-class), 24
nAlignGs-methods (gapSites-class), 24
nAligns (gapSites-class), 24
nAligns,gapSites-method
  (gapSites-class), 24
nAligns-methods (gapSites-class), 24
overlap_genome (gapSites-class), 24
plot_diff (annGapSites-class), 9
plot_diff,annGapSites-method
  (annGapSites-class), 9
plot_diff-methods (annGapSites-class), 9
plot_diff_ranks (alt_X_ranks), 8
plot_diff_ranks,gapSites-method
  (alt_X_ranks), 8
plot_diff_ranks-methods (alt_X_ranks), 8
plotGeneAlignDepth, 32
plotGeneAlignDepth,bamReader-method
  (plotGeneAlignDepth), 32
plotGeneAlignDepth-methods
  (plotGeneAlignDepth), 32
rangeByGeneName, 33
rbamtools, 3
rCodons (xCodons), 46
rCodons,cRanges-method (xCodons), 46
rCodons-methods (xCodons), 46
readCuffGeneFpkm, 34
readExpSet, 35
readMergedBamGaps, 36
readTabledBamGaps, 36
refGenome, 3
rend (gapSites-class), 24
rend,gapSites-method (gapSites-class), 24
rend-methods (gapSites-class), 24
resize_left (trim), 40
resize_left,gapSites-method (trim), 40
INDEX

trypsinCleave-methods (trypsinCleave), 43

uniqueJuncAnn, 44
uniqueJuncAnn, ExpressionSet, refJunctions-method
  (uniqueJuncAnn), 44
uniqueJuncAnn-methods (uniqueJuncAnn), 44

width, cRanges-method (cRanges-class), 16
write.annDNA.tables (gapSites-class), 24
write.annDNA.tables, gapSites, DNAStringSet, character-method
  (gapSites-class), 24
write.annDNA.tables-methods
  (gapSites-class), 24
write.files, 45
write.files, aaGapSites-method
  (aaGapSites-class), 3
write.files, caRanges-method
  (caRanges-class), 13
write.files-methods (write.files), 45

xCodons, 46
xJunc, 47
xJuncStrand, 48