Package 'icetea'

October 16, 2019

Type Package

Title Integrating Cap Enrichment with Transcript Expression Analysis

Version 1.2.0

Description icetea (Integrating Cap Enrichment with Transcript Expression Analysis) provides functions for end-to-end analysis of multiple 5'-profiling methods such as CAGE, RAMPAGE and MAPCap, beginning from raw reads to detection of transcription start sites using replicates. It also allows performing differential TSS detection between group of samples, therefore, integrating the mRNA cap enrichment information with transcript expression analysis.

Depends R (>= 3.5)

Imports stats, utils, methods, graphics, grDevices, ggplot2, GenomicFeatures, ShortRead, BiocParallel, Biostrings, S4Vectors, Rsamtools, BiocGenerics, IRanges, GenomicAlignments, GenomicRanges, rtracklayer, SummarizedExperiment, VariantAnnotation, limma, edgeR, csaw, DESeq2, TxDb.Dmelanogaster.UCSC.dm6.ensGene

Suggests knitr, rmarkdown, Rsubread (>= 1.29.0), testthat

VignetteBuilder knitr

biocViews ImmunoOncology, Transcription, GeneExpression, Sequencing, RNASeq, Transcriptomics, DifferentialExpression

URL https://github.com/vivekbhr/icetea

BugReports https://github.com/vivekbhr/icetea/issues

License GPL-3 + file LICENSE

Encoding UTF-8

RoxygenNote 6.1.1

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

git_branch RELEASE_3_9

git_last_commit 833adaf

git_last_commit_date 2019-05-02

Date/Publication 2019-10-15

Author Vivek Bhardwaj [aut, cre]

Maintainer Vivek Bhardwaj <bhardwaj@ie-freiburg.mpg.de>

R topics documented:

`activeChrs` *Match BAM headers bw files and get active chromosome list (from restrict) (written by Aaron Lun, 12 Dec 2014, copied and modified here)*

Description

Match BAM headers bw files and get active chromosome list (from restrict) (written by Aaron Lun, 12 Dec 2014, copied and modified here)

Usage

```
activeChrs(bam.files, restrict)
```

Arguments

bam.files	Character . bam files to check
restrict	character. Chromosomes to select

Value

Vector of selected chromosomes

---

annotateTSS *Annotate the provided Transcription Start Sites*

Description

This function annotates the provided TSS bed file to provide the number of TSS falling within the genomic features from a given TxDB object. In order to break ties between overlapping features, the function ranks the features by preference. By default, the following order is used: fiveUTR > promoter > intron > coding > spliceSite > threeUTR > intergenic. A custom order of feature ranks can also be provided.

Usage

```
annotateTSS(tssBED, txdb, featureRank = c("fiveUTR", "promoter",
  "intron", "coding", "spliceSite", "threeUTR", "intergenic"),
  plotValue = "number", outFile = NULL)
```

Arguments

tssBED	A bed file with detected TSS/differential TSS coordinates
txdb	A txdb object.
featureRank	A vector with features to use for breaking ties, in decending order of preference (highest to lowest),
plotValue	What values to plot (choose from "number", "percent" or NULL for no plot)
outFile	Output file name. (filename extention would be used to determine type). If outfile not specified, the plot would be retured on the screen

Value

A data.frame with number of TSS falling into each feature

Examples

```
# load a txdb object
library("TxDb.Dmelanogaster.UCSC.dm6.ensGene")
seqlevelsStyle(TxDb.Dmelanogaster.UCSC.dm6.ensGene) <- "ENSEMBL"
# limiting the annotation to X chromosome
seqlevels(TxDb.Dmelanogaster.UCSC.dm6.ensGene) <- "X"

# annotate a given TSS bed file
dir <- system.file("extdata", package = "icetea")
tss <- file.path(dir, "testTSS_merged.bed")
annotations <- annotateTSS(tssBED = tss, TxDb.Dmelanogaster.UCSC.dm6.ensGene,
               plotValue = "number", outFile = "TSS_annot.pdf")
```

check_capSet *Check capset validity*

Description

Check capset validity

Usage

```
check_capSet(object)
```

Arguments

object	capset object

Value

boolean

demultiplexFASTQ *Demultiplex and tag fastq files using sample barcodes*

Description

Demultiplex and tag fastq files using sample barcodes

Usage

```
demultiplexFASTQ(CSobject, outdir, max_mismatch = 0, ncores = 1)

## S4 method for signature 'CapSet'
demultiplexFASTQ(CSobject, outdir, max_mismatch = 0,
  ncores = 1)
```

Arguments

CSobject	CapSet object created using newCapSet function
outdir	character. path to output directory
max_mismatch	integer. maximum allowed mismatches in the sample barcode
ncores	integrer. No. of cores/threads to use

Value

de-multiplxed fastq files corresponding to each barcode. The files are written on disk with the corresponding sample names as specified in the CapSet object

Examples

```
# load a previously saved CapSet object
cs <- exampleCSobject()

# demultiplex allowing one mismatch in sample indexes
dir.create("demult_fastq")
cs <- demultiplexFASTQ(cs, outdir =  "demult_fastq", max_mismatch = 1)
```

`detectDiffTSS` *Detect differentially expressed Transcription Start Sites between two conditions (test)*

Description

Detect differentially expressed Transcription Start Sites between two conditions (test)

Usage

```
detectDiffTSS(fit, testGroup, contGroup, TSSfile = NULL,
  MAplot_fdr = NA)

## S4 method for signature 'DGEGLM'
detectDiffTSS(fit, testGroup, contGroup,
  TSSfile = NULL, MAplot_fdr = NA)

## S4 method for signature 'DESeqDataSet'
detectDiffTSS(fit, testGroup, contGroup,
  MAplot_fdr = NA)
```

Arguments

`fit`	DGEGLM object (output of `fitDiffTSS` command)
`testGroup`	Test group name
`contGroup`	Control group name
`TSSfile`	The TSS .bed file used for `fitDiffTSS` command (if method "edgeR" was used)
`MAplot_fdr`	FDR threshold to mark differentially expressed TSS in MAplot (NA = Don't make an MAplot)

Value

A `GRanges` object containing p-values of differential expression for each TSS.

Examples

```
# before running this
# 1. Create a CapSet object
# 2. de-multiplex the fastqs
# 3. map them
# 4. filter duplicate reads from mapped BAM
```

```
# 5. detect TSS
# 6. fit the diff TSS model.

## Not run:
# load a previously saved DGEGLM object from step 5
csfit <- load("diffTSS_fit.Rdata")
dir <- system.file("extdata", package = "icetea")
# detect differentially expressed TSS between groups (return MA plot)
detectDiffTSS(csfit, testGroup = "mut", controlGroup = "wt",
               tssFile = file.path(dir, "testTSS_merged.bed"), MAplot_fdr = 0.05)


## End(Not run)

## Not run:
# load a previously saved DGEGLM object from step 5
csfit <- load("diffTSS_fit.Rdata")
dir <- system.file("extdata", package = "icetea")
# detect differentially expressed TSS between groups (return MA plot)
detectDiffTSS(csfit, testGroup = "mut", controlGroup = "wt", MAplot_fdr = 0.05)


## End(Not run)
```

`detectTSS` *Detection of Trancription start sites based on local enrichment*

Description

Detection of Trancription start sites based on local enrichment

Usage

```
detectTSS(CSobject, groups, outfile_prefix = NULL, windowSize = 10L,
  sliding = TRUE, foldChange = 2, restrictChr = NULL, ncores = 1)

## S4 method for signature 'CapSet'
detectTSS(CSobject, groups, outfile_prefix = NULL,
  windowSize = 10L, sliding = TRUE, foldChange = 2,
  restrictChr = NULL, ncores = 1)
```

Arguments

`CSobject`	CapSet object created using `newCapSet` function
`groups`	a character vector that contains group name of the sample, for replicate-based TSS calling (see example)
`outfile_prefix`	Output name prefix for the .Rdata file containing window counts, background counts and filtering statistics calculated during TSS detection.
`windowSize`	Size of the window to bin the genome for TSS detection. By default, a window size of 10 is used for binning the genome, however smaller window sizes can

	optionally be provided for higher resolution TSS detection. Note that the background size is set to 200x the window size (2kb for 10bp windows) to calculate local enrichment. Adjacent enriched windows are merged with a distance cutoff, which is the same as window size to get final TSS widths.
sliding	TRUE/FALSE. Indicating whether or not to use sliding windows. The windows are shifted by length which is half of the specified window length.
foldChange	A fold change cutoff of local enrichment to detect the TSS. For samples with usual' amount of starting material and squencing depth (>=5ug starting material, = 5 mil reads/sample), a cut-off of 4-6 fold can be used. For samples with low amount of material or sequencing depth, use a lower cut-off (eg. use 2-fold for samples with 500ng starting material). The final "score" of detected TSS is the mean fold-change of all consecutive windows that passed the foldChange cutoff.
restrictChr	Chromosomes to restrict the analysis to.
ncores	No. of cores/threads to use

Value

.bed files containing TSS position for each group, along with a bed file for consensus (union) TSS sites of all samples.

Examples

```
# before running this
# 1. Create a CapSet object
# 2. de-multiplex the fastqs
# 3. map them
# 4. filter duplicate reads from mapped BAM

# load a previously saved CapSet object
cs <- exampleCSobject()
# detect TSS (samples in same group are treated as replicates)
cs <- detectTSS(cs, groups = rep(c("wt","mut"), each = 2), outfile_prefix = "testTSS",
           foldChange = 6, restrictChr = "X", ncores = 1)
```

---

diffQCplots *Make DESeq2 or edgeR QC plots*

Description

Make DESeq2 or edgeR QC plots

Usage

```
diffQCplots(method, fit, y)
```

Arguments

method	one of "DESeq2" or "edgeR"
fit	output of fitDiffTSS (if method = "edgeR")
y	output of fitDiffTSS (if method = "DESeq2")

exampleCSobject *Create example CapSet object*

Description

Create example CapSet object

Usage

```
exampleCSobject(expMethod = "MAPCap")
```

Arguments

expMethod	Which experiment method to use (options : "RAMPAGE", "MAPCap")

Value

An object of class CapSet

Examples

```
cs <- exampleCSobject(expMethod = "MAPCap")
```

exportTSS *Export the detected TSS from CapSet object as .bed files*

Description

Export the detected TSS from CapSet object as .bed files

Usage

```
exportTSS(CSobject, outfile_prefix, pergroup = FALSE, merged = TRUE)

## S4 method for signature 'CapSet'
exportTSS(CSobject, outfile_prefix, pergroup = FALSE,
  merged = TRUE)
```

Arguments

CSobject	The modified CapSet object after running detectTSS function
outfile_prefix	Prefix (with path) for output .bed files
pergroup	If TRUE, write output per group of samples
merged	If TRUE, write merged bed file (union of all groups)

Value

.bed file(s) containing detected TSS.

Examples

```
# load a previously saved CapSet object
cs <- exampleCSobject()
# export tss
exportTSS(cs, merged = TRUE, outfile_prefix = "testTSS")
```

`filterDuplicates` *Filter PCR-duplicates from mapped files using internal UMIs*

Description

This script considers the read mapping start position and the UMI to determine whether a read is a PCR duplicate. All PCR duplicates are then removed and one entry per read is kept. In case of paired-end reads (MAPCap/RAMPAGE), only one end (R1) is kept after filtering, unless 'keepPairs" is set to TRUE

Usage

```
filterDuplicates(CSobject, outdir, ncores = 1, keepPairs = FALSE)

## S4 method for signature 'CapSet'
filterDuplicates(CSobject, outdir, ncores = 1,
  keepPairs = FALSE)
```

Arguments

`CSobject`	an object of class `CapSet`
`outdir`	character. output directory for filtered BAM files
`ncores`	integer. No. of cores to use
`keepPairs`	logical. indicating whether to keep pairs in the paired-end data. (note: the pairs are treated as independent reads during duplicate removal). Also use keepPairs = TRUE for single-end data.

Value

modified CapSet object with filtering information. Filtered BAM files are saved in 'outdir'.

Examples

```
# before running this
# 1. Create a CapSet object
# 2. de-multiplex the fastqs
# 3. map them

# load a previously saved CapSet object
cs <- exampleCSobject()
# filter duplicate reads from mapped BAM files
dir.create("filtered_bam")
cs <- filterDuplicates(cs, outdir = "filtered_bam")
```

filterDups *Filter PCR-duplicates from BAM file using internal UMIs*

Description

Filter PCR-duplicates from BAM file using internal UMIs

Usage

```
filterDups(bamFile, outFile, keepPairs)
```

Arguments

bamFile	character. Input BAM file
outFile	character. Output (filtered) BAM file
keepPairs	logical. Keep R2 read?

Value

Filtered BAM file, after PCR duplicate removal

fitDiffTSS *Detect differentially expressed Transcription Start Sites between two conditions (fit model)*

Description

Detect differentially expressed Transcription Start Sites between two conditions (fit model)

Usage

```
fitDiffTSS(CSobject, TSSfile = NULL, groups, method = "DESeq2",
  normalization = NULL, normFactors = NULL, outplots = NULL,
  plotRefSample = NA, ncores = 1)

## S4 method for signature 'CapSet'
fitDiffTSS(CSobject, TSSfile = NULL, groups,
  method = "DESeq2", normalization = NULL, normFactors = NULL,
  outplots = NULL, plotRefSample = NA, ncores = 1)
```

Arguments

CSobject	An object of class CapSet
TSSfile	A .bed file with TSS positions to test for differential TSS analysis. If left empty, the union of detected TSS present within the provided CSobject would be plotted.
groups	Character vector indicating the group into which each sample within the CSobject falls. the groups would be use to create a design matrix. As an example, replicates for one condition could be in the same group.

method	Which method to use for differential expression analysis? options are "DESeq2" or "edgeR". If "DESeq2" is chosen, the library size is either estimated via DESeq2 (using "median of ratios") or can be provided via the "normFactors" option below. Setting the "normalization" (below) has no effect in that case.
normalization	A character indicating the type of normalization to perform. Options are "windowTMM", "TMM", "RLE", "upperquartile" or NULL (don't compute normalization factors). If "windowTMM" is chosen, the normalization factors are calculated using the TMM method on 10 kb windows of the genome. "TMM" computes TMM normalization using counts from all the evaluated TSSs. If NULL, the external normalization factors can be used (provided using 'normFactors').
normFactors	external normalization factors (from Spike-Ins, for example).
outplots	Output pdf filename for plots. If provided, the plots for BCV, dispersion and MDS plot is created and saved in this file.
plotRefSample	Name of reference sample to plot for detection of composition bias in the data. Samples could be normalized using one of the provided normalization methods to control for composition bias.
ncores	No. of cores/threads to use

Value

An object of class DGEGLM-class.

Examples

```
# before running this
# 1. Create a CapSet object
# 2. de-multiplex the fastqs
# 3. map them
# 4. filter duplicate reads from mapped BAM
# 5. detect TSS
# 6. fit the diffTSS model
## Not run:
# load a previously saved CapSet object
cs <- exampleCSobject()

# count reads on all TSS (union) and fit a model using replicates within groups
csfit <- fitDiffTSS(cs, groups = rep(c("wt","mut"), each = 2), normalization = "internal",
                    outplots = NULL, plotRefSample = "embryo1")
save(csfit, file = "diffTSS_fit.Rdata")

## End(Not run)
```

---

getBamFlags *Get flags to read from bam*

Description

Get flags to read from bam

Usage

```
getBamFlags(countAll)
```

Arguments

countAll	logical. count all reads?

Value

bamFlags

---

getChromBins *Get chromosome bins from BAM files*

Description

Get chromosome bins from BAM files

Usage

```
getChromBins(bamFiles, restrictChr = NULL, binSize)
```

Arguments

bamFiles	Character. bam files
restrictChr	character. Chromosomes to select
binSize	numeric. Size of bins

Value

GRanges (bins) for both strands

---

getChromWindows *Get chromosome sliding windows from BAM files*

Description

Get chromosome sliding windows from BAM files

Usage

```
getChromWindows(bamFiles, restrictChr = NULL, binSize, stepSize)
```

Arguments

bamFiles	Character vector (bam files)
restrictChr	Chromosomes to select
binSize	Size of bins
stepSize	Size of window slide

Value

GRanges (sliding windows) for both strands

`getGeneCounts` *Get gene-level counts from TSS data*

Description

Get gene-level counts from TSS data

Usage

```
getGeneCounts(CSobject, transcriptGRL, regionAroundTSS = 500,
  outfile = NA, ncores = 1)

## S4 method for signature 'CapSet'
getGeneCounts(CSobject, transcriptGRL,
  regionAroundTSS = 500, outfile = NA, ncores = 1)
```

Arguments

`CSobject`	The `CapSet` object to use.
`transcriptGRL`	A GRangesList object containing transcripts, created using transcriptsBy(txdb)
`regionAroundTSS`	integer, indicating how many bases downstream of TSS to count
`outfile`	character. Tab-separated output file name (if required)
`ncores`	integer. No. of cores/threads to use

Value

data.frame with gene-level counts for all genes in the txdb object

Examples

```
# load a txdb object
library("TxDb.Dmelanogaster.UCSC.dm6.ensGene")
seqlevelsStyle(TxDb.Dmelanogaster.UCSC.dm6.ensGene) <- "ENSEMBL"

# get transcripts by gene (only X chromsome, for simplicity)
seqlevels(TxDb.Dmelanogaster.UCSC.dm6.ensGene) <- "X"
dm6trans <- transcriptsBy(TxDb.Dmelanogaster.UCSC.dm6.ensGene, "gene")

# load a CapSet object
cs <- exampleCSobject()
# get gene counts, counting reads around 500 bp of the TSS
gcounts <- getGeneCounts(cs, dm6trans)
```

getMCparams *Get platform-specific multicore params*

Description

Get platform-specific multicore params

Usage

```
getMCparams(cores)
```

Arguments

cores	integer. No. of cores to use.

Value

BPPARAM object

getNormFactors *Calculate normalization factors from CapSet object*

Description

Calculate normalization factors from CapSet object

Usage

```
getNormFactors(CSobject, features, method = "TMM", ...)

## S4 method for signature 'CapSet'
getNormFactors(CSobject, features, method = "TMM",
  ...)
```

Arguments

CSobject	An object of class CapSet
features	A GRanges-class.object to count the reads on.
method	Method to use for normalization. Options : "TMM","RLE","upperquartile","none"
...	Additional arguments passed to calcNormFactors

Value

Numeric vector of calculated normalization factors.

Examples

```
 # load a txdb object
 library("TxDb.Dmelanogaster.UCSC.dm6.ensGene")
 seqlevelsStyle(TxDb.Dmelanogaster.UCSC.dm6.ensGene) <- "ENSEMBL"

 # get genes (only X chromsome, for simplicity)
 seqlevels(TxDb.Dmelanogaster.UCSC.dm6.ensGene) <- "X"
 dm6genes <- genes(TxDb.Dmelanogaster.UCSC.dm6.ensGene)

# get norm factors by counting reads on genes
cs <- exampleCSobject()
normfacs <- getNormFactors(cs, dm6genes, method = "RLE")
```

---

getranks *Assign feature ranks on a VariantAnnotation output*

Description

Assign feature ranks on a VariantAnnotation output

Usage

```
getranks(x, rank_vec)
```

Arguments

x	output from VariantAnnotation
rank_vec	the pre-set vector of ranks

Value

A vector of ranks of length = length of input features

---

get_newfastq *Get data to create new ShortReadQ object after barcode trimming*

Description

Get data to create new ShortReadQ object after barcode trimming

Usage

```
get_newfastq(type, fq_R1, fq_R2)
```

Arguments

type	character. expType of the CapSet object
fq_R1	character. fastq Read 1
fq_R2	character. fastq Read 2

Value

A list with new R1 and R2 sequence, quality, barcode string and sample id

---

mapCaps *Map the data from 5' profiling techniques*

Description

Map the data from 5' profiling techniques

Usage

```
mapCaps(CSobject, genomeIndex, outdir, externalGTF = NULL, ncores = 1,
  logfile = NULL)

## S4 method for signature 'CapSet'
mapCaps(CSobject, genomeIndex, outdir,
  externalGTF = NULL, ncores = 1, logfile = NULL)
```

Arguments

CSobject	An object of class CapSet
genomeIndex	character. Path to the Subread index file. Should end with the basename of the index.
outdir	character. Output directory path
externalGTF	character. provide external annotation file in 'GTF' format , if present to increase alignment accuracy
ncores	integer. Number of cores/threads to use for mapping.
logfile	character. A log file to write the processing message.

Value

modified CapSet object with mapping information. Mapped and sorted BAM files are saved in 'outdir'.

Examples

```
## Not run:
# before mapping :
# 1. Create a CapSet object
# 2. de-multiplex the fastqs

# load a previously saved CapSet object
cs <- exampleCSobject()

# map the data (not available on windows)
library(Rsubread)
dir.create("bam")
buildindex(basename = "dm6", reference = "/path/to/dm6_genome.fa")
cs <- mapCaps(cs, genomeIndex = "dm6", outdir = "bam", nthreads = 10)
```

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

`newCapSet` *Create a new CapSet object*

Description

The function creates an object of class 'CapSet', used for the TSS analysis. A CapSet object can be created using the the raw, multiplxed fastq files along with the list of sample indexes and corresponding sample names. In case the files are already de-multiplexed or mapped, the CapSet object can also be created using the path to demultiplexed fastq/mapped or filtered BAM files, along with corresponding sample names. In these cases statistics and operations for the missing files would not be possible.

Usage

```
newCapSet(expMethod, fastq_R1 = NULL, fastq_R2 = NULL,
  idxList = NULL, sampleNames, demult_R1 = NA, demult_R2 = NA,
  mapped_file = NA, filtered_file = NA, paired_end = TRUE)
```

Arguments

`expMethod`	experiment method ('CAGE', 'RAMPAGE' or 'MAPCap')
`fastq_R1`	path for Read R1 (or file path for single end reads)
`fastq_R2`	path for Read R2 (for paired end reads)
`idxList`	a vector of index sequences (for demultiplexing)
`sampleNames`	(required) a vector of sample names corresponding to the provided files
`demult_R1`	a vector of file paths for demultiplexed R1 reads
`demult_R2`	a vector of file paths for demultiplexed R2 reads
`mapped_file`	a vector of file paths for mapped BAM files.
`filtered_file`	a vector of file paths for de-duplicated BAM files.
`paired_end`	logical, indiciting whether the data is paired end

Value

An object of class CapSet

Slots

`fastqType` Type of fastq ('single' or 'paired')

`fastq_R1` Path to R1 fastq

`fastq_R2` Path to R1 fastq (for paired-end data)

`expMethod` Name of protocol (RAMPGE or MAPCap)

`sampleInfo` A DataFrame object created using information from `newCapSet` function

`tss_detected` A GRangesList object of detected TSS

Examples

```
# list of barcode IDs
idxlist <- c("CAAGTG", "TTAGCC", "GTGGAA", "TGTGAG")

dir <- system.file("extdata", package="icetea")
# corresponding sample names
fnames <- c("embryo1", "embryo2", "embryo3", "embryo4")

## CapSet object from raw (multiplexed) fastq files
cs <- newCapSet(expMethod = 'MAPCap',
       fastq_R1 = file.path(dir, 'mapcap_test_R1.fastq.gz'),
       fastq_R2 = file.path(dir, 'mapcap_test_R2.fastq.gz'),
       idxList = idxlist,
       sampleNames = fnames)

## CapSet object from mapped BAM files
bams <- list.files(file.path(dir, 'bam'), pattern = '.bam$', full.names = TRUE)
cs <- newCapSet(expMethod = 'MAPCap',
       mapped_file = bams,
       sampleNames = fnames)
```

---

numReadsInBed *Count the number of reads in a given GRanges*

Description

Count the number of reads in a given GRanges

Usage

```
numReadsInBed(regions, bams = NA, countall = FALSE)
```

Arguments

regions	The GRanges object to count reads in.
bams	character. path to bam files from where the reads have to be counted
countall	logical. whether to keep both reads of paired-end data

Value

Total counts within given ranges per BAM file.

plotPrecision *Plotprecision background script*

Description

Plotprecision background script

Usage

```
plotPrecision(ref, tssData, distCut)
```

Arguments

ref	GRanges. reference ranges to compare the precision with.
tssData	GRangesList object with TSS detected per sample
distCut	integer. max distance cutoff

Value

ggplot object

plotReadStats *Plot read statistics from the CapSet object*

Description

Plot read statistics from the CapSet object

Usage

```
plotReadStats(CSobject, plotType = "dodge", plotValue = "numbers",
  outFile = NULL)

## S4 method for signature 'CapSet'
plotReadStats(CSobject, plotType = "dodge",
  plotValue = "numbers", outFile = NULL)
```

Arguments

CSobject	The CapSet object
plotType	character. The type of plot to make. Choose from "stack" or "dodge" for either a stacked barchart, or a bar chart with "dodged" positions (analogous to ggplot)
plotValue	character. What values to plot. Choose from "numbers" or "proportions". If "proportions" is selected, the proportion of reads w.r.t total demultiplexed reads per sample would be plotted
outFile	character. Output file name. (filename extention would be used to determine type). If outfile not specified, the plot would be retured on the screen

Value

A ggplot object, or a file. Plot showing the number/proportion of reads in each category, per sample

Examples

```
# load a previously saved CapSet object
cs <- exampleCSobject()
plotReadStats(cs, plotType = "dodge", plotValue = "numbers", outFile = "test_numbers.pdf")
```

---

plotTSSprecision *Compare the precision of TSS detection between multiple samples*

Description

Plot precision of TSS detection from multiple samples (bed files) with respect to a given reference annotation.

Plot precision of TSS detection from multiple samples present within a CapSet object, with respect to a given reference annotation.

Usage

```
plotTSSprecision(reference, detectedTSS, distanceCutoff = 500,
  outFile = NULL, ...)

## S4 method for signature 'GRanges,character'
plotTSSprecision(reference, detectedTSS,
  distanceCutoff = 500, outFile = NULL, sampleNames)

## S4 method for signature 'GRanges,CapSet'
plotTSSprecision(reference, detectedTSS,
  distanceCutoff = 500, outFile = NULL, ...)
```

Arguments

reference	Reference Transcrips/Genes as a GRanges object
detectedTSS	Either a CapSet object with TSS information (after running detectTSS) or a character vector with paths to the BED files containing detcted TSSs
distanceCutoff	integer. Maximum distance (in base pairs) from reference TSS to plot
outFile	character. Output file name (filename extention would be used to determine type) If outfile not specified, the plot would be returned on the screen
...	Additional arguments
sampleNames	character. Labels for input samples (in the same order as the input bed files)

Value

A ggplot object, or a file. Plot showing perent of TSS detected per sample with respect to their cumulative distance to TSS of the provided reference

Examples

```
# load a txdb object
library("TxDb.Dmelanogaster.UCSC.dm6.ensGene")
seqlevelsStyle(TxDb.Dmelanogaster.UCSC.dm6.ensGene) <- "ENSEMBL"
transcripts <- transcripts(TxDb.Dmelanogaster.UCSC.dm6.ensGene)

# Plotting the precision using a pre computed set of TSS (.bed files) :

tssfile <- system.file("extdata", "testTSS_merged.bed", package = "icetea")
plotTSSprecision(reference = transcripts, detectedTSS = tssfile,
                sampleNames = "testTSS", distanceCutoff = 500,
                outFile = "TSS_detection_precision.png")

## Plotting the precision using a CapSet object :

library("TxDb.Dmelanogaster.UCSC.dm6.ensGene")
seqlevelsStyle(TxDb.Dmelanogaster.UCSC.dm6.ensGene) <- "ENSEMBL"
# only use chrX to make the analysis faster
seqlevels(TxDb.Dmelanogaster.UCSC.dm6.ensGene) <- "X"
transcripts <- transcripts(TxDb.Dmelanogaster.UCSC.dm6.ensGene)

# load a previously saved CapSet object
cs <- exampleCSobject()
# plot
plotTSSprecision(reference = transcripts, detectedTSS = cs,
                 outFile = "TSS_detection_precision.png")
```

`ResizeReads` *preprocess reads to count only 5' overlaps*

Description

preprocess reads to count only 5' overlaps

Usage

```
ResizeReads(reads, width = 1, fix = "start")
```

Arguments

`reads`	GAlignment object to resize
`width`	integer. New read length
`fix`	character. 'Start' for 5'

Value

Resized reads as GRanges

`sampleInfo` *Retrieve and replace sample information of a CapSet object*

Description

Retrieve and replace sample information of a CapSet object

Usage

```
sampleInfo(object, ...)

sampleInfo(object, ...) <- value

## S4 method for signature 'CapSet'
sampleInfo(object)

## S4 replacement method for signature 'CapSet'
sampleInfo(object) <- value
```

Arguments

`object`	The `CapSet` object
`...`	Additional options
`value`	Replacement DataFrame object

Value

sample information data.frame

Examples

```
# load a previously saved CapSet object
cs <- exampleCSobject()
# get sampleinfo
si <- sampleInfo(cs)
# modify
si$samples <- paste0("sample_", seq_along(1:nrow(si)) )
# replace
sampleInfo(cs) <- si
```

`splitBAM_byIndex` *Split the composite BAM file using internal indexes (MAPCap)*

Description

Split the composite BAM file using internal indexes (MAPCap)

Usage

```
splitBAM_byIndex(bamFile, index_list, outfile_list, max_mismatch = 0,
  ncores = 1)
```

Arguments

bamFile	character. Path to a mapped BAM file
index_list	character. A list of indexes for splitting
outfile_list	character. A list of output file names (with order corresponding to that of index_list)
max_mismatch	integer. No. of mismatches allowed in index (maxium 1 recommended)
ncores	integer. Number of cores to use for parallel processing

Value

Filtered files

Examples

```
bam <- system.file("extdata", "bam/embryo1.bam", package = "icetea")
splitBAM_byIndex(bamFile = bam,
                 index_list = c("CAAGTG", "CAAGTT"),
                 outfile_list = c("test_filt1.bam","test_filt2.bam"),
                 ncores = 1)
```

---

splitBAM_byRepindex *Split the composite BAM file using replicate indexes (MAPCap data)*

Description

Split the composite BAM file using replicate indexes (MAPCap data)

Usage

```
splitBAM_byRepindex(bamFile, outfile_prefix, ncores = 1)
```

Arguments

bamFile	character. Path to a mapped BAM file
outfile_prefix	character. prefix for output file (replicates IDs will be added as RR/YY)
ncores	integer. Number of cores to use for parallel processing

Value

Filtered files by replicate Index

Examples

```
bam <- system.file("extdata", "bam/embryo1.bam", package = "icetea")
splitBAM_byRepindex(bamFile = bam, outfile_prefix = "testSplit", ncores = 1)
```

`splitranks` *Get features with the best rank for each TSS*

Description

Get features with the best rank for each TSS

Usage

```
splitranks(x)
```

Arguments

`x`	output of getranks

Value

A data frame with counts

`split_fastq` *Split paired-end fastq by barcodes*

Description

Split paired-end fastq by barcodes

Usage

```
split_fastq(expType, idx_name, outfile_R1, outfile_R2, fastq_R1, fastq_R2,
  max_mismatch)
```

Arguments

`expType`	character. experiment type (RAMPAGE, MAPCap or CAGE)
`idx_name`	character. barcode ID
`outfile_R1`	character. output fastq file : Read 1
`outfile_R2`	character. output fastq file : Read 2
`fastq_R1`	character. input fastq file : Read 1
`fastq_R2`	character. input fastq file : Read 2
`max_mismatch`	integer. max allowed mismatches

Value

kept reads corresponding to each barcode.

`strandBinCounts` *Perform stranded Bin counts*

Description

Perform stranded Bin counts

Usage

```
strandBinCounts(bam.files, restrictChrs, bam_param, bp_param, window_size,
  sliding = FALSE)
```

Arguments

`bam.files`	character vector. BAM files to use
`restrictChrs`	character vector. chromosomes to use
`bam_param`	ScanBAMParams
`bp_param`	BPPARAM
`window_size`	integer. size of window to use
`sliding`	logical. perform sliding window counts?

Value

RangedSE object with forward and reverse strand counts

Index