Package ‘diffloop’

June 24, 2016

Type Package

Title Differential DNA loop calling from ChIA-PET data

Version 1.0.2

Date 2016-04-30

Maintainer Caleb Lareau <caleblareau@g.harvard.edu>

Author Caleb Lareau [aut, cre], Martin Aryee [aut]

Description A suite of tools for subsetting, visualizing, annotating, and statistically analyzing the results of one or more ChIA-PET experiments.

Imports methods, GenomicRanges, foreach, plyr, dplyr, reshape2, ggplot2, matrixStats, Sushi, edgeR, locfit, statmod, biomaRt, GenomeInfoDb, S4Vectors, IRanges, grDevices, graphics, stats, utils, Biobase, readr

License MIT + file LICENSE

LazyData TRUE

Suggests diffloopdata, knitr, rmarkdown, testthat

VignetteBuilder knitr

RoxygenNote 5.0.1

Collate 'diffloop.R' 'diffloop-class.R' 'data.R' 'core.R' 'pipeline_io.R' 'loopFunctions.R' 'sugar.R' 'union.R' 'annotation.R' 'assoc.R' 'colData.R' 'plotting.R'

biocViews Preprocessing, QualityControl, Visualization, DataImport, DataRepresentation, GO

NeedsCompilation no

R topics documented:

addchr — *Add 'chr' to GRanges seqnames*

Description

`addchr` takes a loops object or GRanges object and simply adds 'chr' to seqnames

Usage

```
addchr(dlo)

## S4 method for signature 'loops'
addchr(dlo)

## S4 method for signature 'GRanges'
addchr(dlo)
```

Arguments

`dlo`	A loops object or GRanges object

Details

Often times, performing functions on GRanges objects can go awry if the seqnames are systematically different. A common example of this is when some GRanges objects has the format of 'chr1' while the other has '1'. We can add 'chr' to the first object

Value

An identical loops object or GRanges object 'chr' added

Examples

```
library(GenomicRanges)
regA <- GRanges(c('1'),ranges=IRanges(c(36200000),c(36300000)))
addchr(regA)
regA
rmchr(regA)
regA
```

annotateAnchors *Add meta data column to anchors based on .bed file*

Description

annotateAnchors adds a logical variable to meta data columns in the anchors based on a GRanges object of features' genomic coordinates

Usage

```
annotateAnchors(dlo, features, featureName, maxgap)

## S4 method for signature 'loops,GRanges,character,missing'
annotateAnchors(dlo, features,
  featureName, maxgap = 1000)

## S4 method for signature 'loops,GRanges,character,numeric'
annotateAnchors(dlo, features,
  featureName, maxgap)
```

Arguments

dlo	A loops object whose anchors will be annotated
features	A Granges object corresponding to locations of interest
featureName	A string that will be the mcol name in anchors
maxgap	A value of max permissible gap between a feature and anchor

Details

This function adds column of TRUE/FALSE values on the loops object anchors whether a feature is observed nearby in `features`. The name of this column that will be in the anchors GRanges object is specified by a user defined string `featureName`. Gap tolerance between a feature and an anchor is specified by `maxgap`, where the default is 1,000bp.

Value

A loops object with new meta data column in anchors

Examples

```
# Annotate whether anchors are near a gene body; within 1kb
rda<-paste(system.file('rda',package='diffloop'),'loops.small.rda',sep='/')
load(rda)
gb <-getHumanGenes()
loops.small <- annotateAnchors(loops.small,gb,'nearGeneBody')

# Adding close to gene bodies with no gap tolerance
```

```
loops.small <- annotateAnchors(loops.small,gb,'inGeneBody',0)
```

---

annotateLoops — *Annotate loops as Enhancer-Promoter or CTCF-CTCF*

Description

annotateLoops adds a column to the rowData slot of a loops object categorizing loops as either e-p (enhancer-promoter), ctcf (CTCF-CTCF) or none (no biological annotation). If both ctcf and e-p, then categorized as e-p.

Usage

```
annotateLoops(lto, ctcf, enhancer, promoter)

## S4 method for signature 'loops,GRanges,GRanges,GRanges'
annotateLoops(lto, ctcf, enhancer,
  promoter)
```

Arguments

lto	A loops object whose loops will be annotated
ctcf	GRanges object corresponding to locations of CTCF peaks
enhancer	GRanges object corresponding to locations of enhancer peaks
promoter	GRanges object corresponding to locations of promoter regions

Details

Function annotates loops where both anchors are near CTCF peaks or where one anchor is near an enhancer and the other near a promoter. Consider using functions addchr, rmchr, bedToGRanges, and padGRanges when setting up the 3 GRanges inputs. Provide a blank GRanges objects to ignore classification for one set.

Value

A loops object with an additional row 'loop.type' in the rowData slot

Examples

```
rda<-paste(system.file('rda',package='diffloop'),'loops.small.rda',sep='/')
load(rda)
ctcf_j <- system.file('extdata','Jurkat_CTCF_chr1.narrowPeak',package='diffloop')
ctcf <- rmchr(padGRanges(bedToGRanges(ctcf_j), pad = 1000))
h3k27ac_j <- system.file('extdata','Jurkat_H3K27ac_chr1.narrowPeak',package='diffloop')
h3k27ac <- rmchr(padGRanges(bedToGRanges(h3k27ac_j), pad = 1000))
promoter <- padGRanges(getHumanTSS(c('1')), pad = 1000)
jn <- loops.small[,c(1,2,5,6)]
```

```
assoc_jn <- quickAssoc(jn)
assoc_jn <- removeSelfLoops(assoc_jn)
annotated_jn <- annotateLoops(assoc_jn, ctcf, h3k27ac, promoter)
```

`bedToGRanges` *Read a file and make a GRanges object*

Description

`bedToGRanges` takes a string corresponding to a file and creates a GRanges object, retaining meta-data

Usage

```
bedToGRanges(file)

## S4 method for signature 'character'
bedToGRanges(file)
```

Arguments

`file`	A string specifiying .bed file location

Details

Useful function to read in a .bed file to create a GRanges object where the meta-data is presevered. Useful for later functions like `annotateAnchors`

Value

A GRanges object

Examples

```
#Read in CTCF Jurkat peaks in
ctcf_j <- system.file('extdata','Jurkat_CTCF_chr1.narrowPeak',package = 'diffloop')
ctcf <- bedToGRanges(ctcf_j)
```

`calcLDSizeFactors` *Compute normalizing factors for each sample*

Description

`calcLDSizeFactors` takes a loops object computes size factors based for each sample

Usage

```
calcLDSizeFactors(dlo)

## S4 method for signature 'loops'
calcLDSizeFactors(dlo)
```

Arguments

`dlo`	A loops object with unnormalized size factors

Details

This function updates the `loops` object with new `sizeFactor` values for each sample in the `colData` slot using a method identical to that employed in `DESeq2`.

Value

A loops object with new size factors in `colData`

Examples

```
# Computing normalizing factors from the full ChIA-PET Data
rda<-paste(system.file('rda',package='diffloop'),'loops.small.rda',sep='/')
load(rda)
loops.small <- calcLDSizeFactors(loops.small)
```

`diffloop` *diffloop: A package for differential DNA loop calling from ChIA-PET data*

Description

The diffloop package contains a suite of tools and S4 data objects to efficiently facilitate the analysis of ChIA-PET datasets. Key features include differential loop calling, visualization of looping in regions, quality-control metrics, and principal component analysis across experiments.

diffloop classes

Three classes mostly comprise the methodology in `diffloop`. First, `loops` is a basic structure that contains one or more ChIA-PET experiments, `loopfit` links an `edgeR` fit to a `loops` and currently has little functionality except for generating another `loops` object where per-loop summary statistics are added.

`dim,loops-method` *See dimensions of loops object*

Description

See dimensions of loops object

Usage

```
## S4 method for signature 'loops'
dim(x)
```

Arguments

`x`	A loops object

Value

A data.frame of dimensions of the loops object, including number of anchors, interactions, samples, and column data attributes

---

`featureTest` *Combined association test for all loops in a defined region*

Description

`featureTest` takes a `loops` and genomic coordinates of regions and computes combined significance metrics for each region using the Simes procedure

Usage

```
featureTest(x, features)

## S4 method for signature 'loops,GRanges'
featureTest(x, features)
```

Arguments

`x`	A loops object
`features`	A GRanges object defining regions for a combined test

Details

This function returns a data.frame sorted by FDR of each region. Assumes the region name is specified in the GRanges object with `id` column. Each feature is a one row in the GRanges object. The combined significance measure per feature is computed via the Simes method for intrachromosomal loops where at least one anchor from the loop overlaps with the region of interest.

Value

A data.frame sorted by FDR

Examples

```
# Human genes chromosome 1 regional association
rda<-paste(system.file('rda',package='diffloop'),'loops.small.rda',sep='/')
load(rda)
jpn_loopfit <- loopFit(loops.small)
# Differential loop calling between naive and jurkat
assoc_jn <- loopTest(jpn_loopfit, coef = 2)
# Gene based association
sw_jn <- featureTest(assoc_jn, getHumanGenes(c('1')))
```

`filterLoops` *Filter loops*

Description

`filterLoops` filters out loops that aren't wide, aren't prevalent within samples or prevalent between samples

Usage

```
filterLoops(dlo, width = 0, nreplicates = 0, nsamples = 1)

## S4 method for signature 'ANY'
filterLoops(dlo, width = 0, nreplicates = 0, nsamples = 1)
```

Arguments

`dlo`	A loops object
`width`	Minimum loop width
`nreplicates`	Minimum number of counts per loop
`nsamples`	Minimum number of samples per loop per counts

Details

Function that restricts loops in a loops object. `width` specifies the minimum width between anchors. Default is zero. `nreplicates` restricts loops to at least this specified amount of counts is present in at least one sample. Instead of `nreplicates` being present in only one sample, `nsamples` specifies how many individual samples that a loop must have `nreplicates` in to be included after filtering.

Value

A loops object

Examples

```
rda<-paste(system.file('rda',package='diffloop'),'loops.small.rda',sep='/')
load(rda)
# Restrict loops to > 5kb width
filtered.jpn1 <- filterLoops(loops.small, 5000, 0, 0)
# Restrict loops to > 5kb width and have >= 3 replicates in >= 1 sample
filtered.jpn2 <- filterLoops(loops.small, 5000, 3, 1)
# Restrict loops to > 10kb width and have >= 3 replicates in >= 2 samples
filtered.jpn3 <- filterLoops(loops.small, 10000, 3, 2)
```

`geneinfo` *Human/mouse exon locations*

Description

A dataframe used for plotting annotation for human and mouse. Each loaded .rda has the same variable called "geneinfo" (so don't co-load these), but the files differ by an m orh

Usage

```
geneinfo
```

Format

A GRanges object

chrom Chromosomes without "chr"

start exon start location

stop exon end location

gene Gene Name

score dummy column there for sushi

strand +1 or -1 to indicate side of DNA ...

Value

A data.frame

Source

biomaRt July 2015 stable build

`getHumanGenes` *Get protein coding gene regions*

Description

`getHumanGenes` returns a `GRanges` object of all protein coding genes genome-wide or within specified chromosomes

Usage

```
getHumanGenes(chr, cache = TRUE)

## S4 method for signature 'missing'
getHumanGenes(chr, cache = TRUE)

## S4 method for signature 'character'
getHumanGenes(chr, cache = TRUE)
```

Arguments

`chr`	A vector of chromosomes
`cache`	logic variable (default = TRUE) to use genes from July.2015 freeze

Details

This function returns a `GRanges` object with the coordinates and gene IDs of all protein coding genes either genome-wide (by default) orspecified within a particular chromosome.

Value

A GRanges object

Examples

```
# Grab all protein coding gene locations genome-wide
pc.genes <- getHumanGenes()
# Grab all protein coding gene loctions on chromosome 1
chr1 <- getHumanGenes(c('1'))
```

getHumanTSS *Get Human Transcription Start Sites*

Description

getHumanTSS returns a GRanges object of all transcription start sites for humans

Usage

```
getHumanTSS(chr, cache = TRUE)

## S4 method for signature 'missing'
getHumanTSS(chr, cache = TRUE)

## S4 method for signature 'character'
getHumanTSS(chr, cache = TRUE)
```

Arguments

chr	Specifies what chromosomes are desired for the TSS
cache	logic variable (default = TRUE) to use TSS from July.2015 freeze

Details

This function returns a GRanges object with the coordinates and gene TSS. The start and end of the IRanges slot will be the same number, so consider using the padGRanges function after calling this function.

Value

A GRanges object

Examples

```
# Grab all transition start sites genome-wide
human.TSS <- getHumanTSS()
```

`head,loops-method` *Extract first part of loops object*

Description

Extract first part of loops object

Usage

```
## S4 method for signature 'loops'
head(x, n = 6, ...)
```

Arguments

`x`	A loops object
`n`	Number of lines to view
`...`	Other non-essential params

Value

A loops object

`human.genes` *Human protein coding genes*

Description

A GRanges object with the human protein-coding genes

Usage

```
human.genes
```

Format

A GRanges object

seqnames Chromosomes without "chr"

ranges start/end loci

strand not specified ('*' everywhere)

id Gene Name ...

Value

A GRanges object

Source

biomaRt July 2015 stable build

`human.TSS` *Human 60k+ transcription start sites*

Description

A GRanges object with all loci of transcription start sites

Usage

```
human.TSS
```

Format

A GRanges object

seqnames Chromosomes without "chr"

ranges start/end loci are same

strand not specified ('*' everywhere)

id Gene Name ...

Value

A GRanges object

Source

biomaRt July 2015 stable build

`interchromosomal` *Loops between chromosomes*

Description

`interchromosomal` restricts loops to those where anchors are observed on different chromosomes

Usage

```
interchromosomal(dlo)

## S4 method for signature 'loops'
interchromosomal(dlo)
```

Arguments

`dlo`	A loops object

Details

This function subsets the `loops` object into only those loops that have anchors on different chromosomes

Value

A loops object with all loops on different chromosomes

Examples

```
rda<-paste(system.file('rda',package='diffloop'),'loops.small.rda',sep='/')
load(rda)

# Compute number of interactions on same chromosome
dim(intrachromosomal(loops.small))
samechromo <- intrachromosomal(loops.small)

# Compute number of interactions on same chromosome
# dim(interchromosomal(loops.small))
# This will throw and error since the toy only has intrachromosomal loops
```

`intrachromosomal` *Loops within chromosomes*

Description

`intrachromosomal` restricts interactions to those where anchors are observed on the same chromosomes

Usage

```
intrachromosomal(dlo)

## S4 method for signature 'loops'
intrachromosomal(dlo)
```

Arguments

`dlo`	A loops object

Details

This function subsets the `loops` object into only those interactions that have both anchors on the same chromosome

Value

A loops object where all loops are on the same chromosome.

Examples

```
rda<-paste(system.file('rda',package='diffloop'),'loops.small.rda',sep='/')
load(rda)

# Compute number of interactions on same chromosome
dim(intrachromosomal(loops.small))
samechromo <- intrachromosomal(loops.small)
```

`loopFit` *Fit model for association testing*

Description

`loopFit` takes a `loops` object and prepares it for the `loopTest` function.

Usage

```
loopFit(y, design, method = "QLF")

## S4 method for signature 'loops,missing,missing'
loopFit(y, design, method = "QLF")

## S4 method for signature 'loops,matrix,missing'
loopFit(y, design, method = "QLF")
```

Arguments

`y`	A loops object for association
`design`	A design matrix (optional)
`method`	Specifies association; currently only 'QLF' is supported

Details

This function returns a `loopfit` object, which combines the `loops` object in the input with a `DGEGLM` object that is the normal output of an `edgeR glmQLFit`. To set up a different design matrix, pass that parameter through the function. Otherwise, the default is to generate a new matrix from `loops@colData$groups`. Currently, 'QLF' is the only supported method, but new association tests may be added in later developments

Value

A loopfit object

Examples

```
# Differential loop fit
rda<-paste(system.file('rda',package='diffloop'),'loops.small.rda',sep='/')
load(rda)
jpn_loopfit <- loopFit(loops.small)
# Differential loop calling between naive and jurkat
assoc_jn <- loopTest(jpn_loopfit, coef = 2)
```

`loopfit-class` *A class to represent ChIA-PET interaction data and an edgeR fit.*

Description

A class to represent ChIA-PET interaction data and an edgeR fit.

Slots

`loops` A loops object with anchors, interactions, counts, colData, and rowData

`fit` An edgeR fit from running the loopFit function

`loopGenes` *Determine genes contained within loops*

Description

`loopGenes` determines all gene bodies partially or fully contained in a loop.

Usage

```
loopGenes(dlo, genesGR)

## S4 method for signature 'loops,GRanges'
loopGenes(dlo, genesGR)
```

Arguments

`dlo`	A loops object
`genesGR`	A GRanges object of genes with mcol 'id'

Details

Function that annotates all loops. 'NA' if looping between chromosomes. Otherwise, all gene names that are contained within a loop. 'None' if no genes are in the loop body. If there are multiple, the function returns a comma separated list. The length of the object returned by this function should be the same length as the number of rows in the `loops` slot.

Value

A matrix of comma separated gene names

Examples

```
# Determine the genes housed in the loops from our example
genes <- getHumanGenes()
rda<-paste(system.file('rda',package='diffloop'),'loops.small.rda',sep='/')
load(rda)
loops.small <- loopGenes(loops.small,genes)
```

`loopMetrics` *Types of loops*

Description

`loopMetrics` counts number of loops for each sample and returns whether they are single, self, unique, or none

Usage

```
loopMetrics(dlo)

## S4 method for signature 'loops'
loopMetrics(dlo)
```

Arguments

`dlo`	A loops object

Details

This function shows the number of loops for each sample based on four types. Single refers to having only one anchor for a the loop whereas none has no unique anchors. If using the `loopsMake` pipeline, only self and unique loops will be observed when running this function

Value

A data.frame

Examples

```
# Return loop metrics for number of each type for each sample
rda<-paste(system.file('rda',package='diffloop'),'loops.small.rda',sep='/')
load(rda)
loopMetrics(loops.small)
```

`loopPlot` *Visualize looping*

Description

`loopPlot` takes a `loops` object and a `GRanges` objectand shows all loops in region (where both anchors are present)

Usage

```
loopPlot(x, y, organism = "h", geneinfo = "NA", colorLoops = FALSE,
  cache = TRUE)

## S4 method for signature 'loops,GRanges'
loopPlot(x, y, organism = "h", geneinfo = "NA",
  colorLoops = FALSE, cache = TRUE)
```

Arguments

`x`	A loops object
`y`	A GRanges object containing region of interest
`organism`	'h' for human or 'm' for mouse supported
`geneinfo`	A data.frame manually specifying annotation (see Examples)
`colorLoops`	Differentiates loops based on loop.type in loops object
`cache`	logic variable (default = TRUE) to use gene annotation from July.2015 freeze

Details

Basic plot function shows the looping in each sample. The intensity of the color is proportional to the number of counts observed for the particular loop relative to the other loops in the entire plot. If colorLoops is specified at TRUE, then the x object must be loops and it must have a loop.type column which can be generated from the `annotateLoops` function. Blue loops are CTCF loops; black are none; red are enhancer-promoter loops.

Value

A plot object

Examples

```
# Print loops in region chr1:36000000-36300000
library(GenomicRanges)
rda<-paste(system.file('rda',package='diffloop'),'loops.small.rda',sep='/')
load(rda)
regA <- GRanges(c('1'),IRanges(start=c(36000000),end=c(36300000)))
plot1 <- subsetRegion(loops.small, regA)
#Example of \code{geneinfo} table
```

```
geneinfo <- data.frame(1,359345,359681,'RP5-8572K21.15','.',-1)
names(geneinfo) <- c('chrom','start','stop','gene','strand')
```

`loops-class` *A class to represent ChIA-PET interaction data and annotations*

Description

A class to represent ChIA-PET interaction data and annotations

Slots

`anchors` A GRanges object describing loop anchor locations

`interactions` A matrix. Each row is an interaction between two anchors

`counts` A matrix with the number paired-end reads per loop per sample

`colData` A data.frame with features (colums) for each sample (rows)

`rowData` A data.frame with features (columns) for each loop (rows)

`loops.small` *chr1:36000000-36300000 loops*

Description

A loops object containing unique 108 loops with 27 anchors for 6 samples and corresponding colData/rowData

Usage

```
loops.small
```

Format

A small loops object

anchors GRanges object of anchor locations

loops indexes of interactions

samples Two replicates each of jurkat, naive, and primed cells

colData Groups identifying cell type and unnormalized sizeFactors

rowData Base initialization with only loopWidth values ...

Value

A loops object

Source

subsetRegion(loops,GRanges(c('1'),IRanges(c(36000000),c(36300000))))

loopsMake *Read preprocessed ChiA-PET data*

Description

loopsMake reads in a data directory created by the dnaloop preprocessing pipeline and returns a loops object

Usage

```
loopsMake(beddir, samples = NA, mergegap = 0, type = "all")

## S4 method for signature 'ANY'
loopsMake(beddir, samples = NA, mergegap = 0,
  type = "all")
```

Arguments

beddir	A string. The preprocessed data directory
samples	A character vector. Optional list of samples to read in
mergegap	An integer value of the radius to merge anchors; default 0
type	Specificies 'intra', 'inter', or 'all' looping. Default 'all'

Details

This function reads in preprocessed ChIA-PET data produced by the dnaloop preprocessing pipeline. The preprocessed directory contains one subdirectory per sample. The samples argument specifies which samples are read. if samples is not specified all samples will be read. type restricts loops whether they are on the same 'inter' or different 'intra' chormosome. Default is 'all'

Value

A loops object

Examples

```
# Reading in all samples, no mergegap, all loops
bd<- system.file('extdata', 'esc_jurkat', package='diffloopdata')
# loops <- loopsMake(bd) #standard call

# Reading in a subset of samples, 1kb mergegap, only intrachromosomal
# looping
samples <- c('naive_esc_1', 'naive_esc_2')
naive.intra <- loopsMake(bd, samples, 1000, 'intra')
```

loopsSubset *Subset two difloop objects*

Description

`loopsSubset` takes the interactions and anchors present in dlo1 and uses the counts and samples from dlo2.

Usage

```
loopsSubset(dlo1, dlo2)

## S4 method for signature 'loops,loops'
loopsSubset(dlo1, dlo2)
```

Arguments

`dlo1`	A loops object
`dlo2`	A loops object

Details

This function plays nice with `union` to ensure counts are correct after taking the union of two loops objects. The subset function simply returns the anchors and interactions of dlo1 and the counts and colData of dlo2.

Value

A loops obect

Examples

```
# divide and recombine samples
rda<-paste(system.file('rda',package='diffloop'),'loops.small.rda',sep='/')
load(rda)
naive <- loops.small[,1:2]
primed <- loops.small[,3:4]
np <- union(naive, primed)
# Subset from full to get correct counts
c.np <- loopsSubset(np, loops.small)
```

`loopTest` *Differential Loop Calling*

Description

`loopTest` takes a `loopfit` object from the `loopFit` function and creates a `loops` object with additional columns in the `rowData`

Usage

```
loopTest(y, coef = 2, contrast, method = "QLF")

## S4 method for signature 'loopfit,missing,missing,missing'
loopTest(y, coef = 2, contrast,
  method = "QLF")

## S4 method for signature 'loopfit,numeric,missing,missing'
loopTest(y, coef = 2, contrast,
  method = "QLF")

## S4 method for signature 'loopfit,missing,numeric,missing'
loopTest(y, coef = 2, contrast,
  method = "QLF")
```

Arguments

`y`	A loopfit object for association
`coef`	Specifies coefficient of design matrix
`contrast`	Specifies comparison of groups from design matrix
`method`	Specifies association method; only QLF is currently supported

Details

This function returns a `loops` object, which contains the results from an association in the rowData slot. The default association is using coefficient 2 from the model matrix (e.g. good for pair comparisons) but the user may specify a different coefficient. Currently, 'QLF' is the only supported method, but new features may be added in later developments. Users may also specify the contrast between the columns in the design matrix as used in `edgeR`.

Value

A loops object with additional columns in rowData

Examples

```
# Differential loop fit
rda<-paste(system.file('rda',package='diffloop'),'loops.small.rda',sep='/')
load(rda)
jpn_loopfit <- loopFit(loops.small)
# Differential loop calling between naive and jurkat
assoc_jn <- loopTest(jpn_loopfit, coef = 2)
```

`loopWidth` *Loop widths*

Description

`loopWidth` returns the width of a loop, which is defined as the distance between the anchors containing a loop

Usage

```
loopWidth(dlo)

## S4 method for signature 'loops'
loopWidth(dlo)
```

Arguments

`dlo`	A loops object

Details

This function returns a positive integer value of the number of basepairs that separate two loops. If they are on separate chromosomes, it still returns a value, but it will be non-sensical, so consider subsetting to only intrachromosomal loops. Also, self-loops will return a postive number that is the inter-anchor width. These loops should be handled using the removeSelfLoops() function.

Value

An integer vector

Examples

```
# Return the width for loops
rda<-paste(system.file('rda',package='diffloop'),'loops.small.rda',sep='/')
load(rda)
w <- loopWidth(loops.small)
```

`mergeAnchors` *Combine nearby anchors into one peak*

Description

`mergeAnchors` combines anchors that are within a user-defined radius

Usage

```
mergeAnchors(dlo, mergegap, selfloops = FALSE)

## S4 method for signature 'loops,numeric,missing'
mergeAnchors(dlo, mergegap,
  selfloops = FALSE)

## S4 method for signature 'loops,numeric,logical'
mergeAnchors(dlo, mergegap,
  selfloops = FALSE)
```

Arguments

`dlo`	A loops object whose anchors will be merged
`mergegap`	An integer value of the bp between anchors to be merged
`selfloops`	A logical value to either retain (T) or remove (F) resulting self-loops after merging anchors

Details

This function takes a loops object and combines nearby anchors, up to a distance specified by the `mergegap`. This likely will cause self loops to form (loop where the left and right anchor are the same), which can either be removed (by default) or retained with `selfloops`

Value

A loops object

Examples

```
# Merge anchors within 1kb of each other, keeping self loops
rda<-paste(system.file('rda',package='diffloop'),'loops.small.rda',sep='/')
load(rda)
m1kb <- mergeAnchors(loops.small, 1000, FALSE)

# Merge anchors within 1kb of each other, removing self loops by default
m1kb_unique <- mergeAnchors(loops.small, 1000)
```

numAnchors *Get number of anchors in each sample*

Description

`numAnchors` takes a `loops` object and a summarizes the number of anchors that support all the interactions (count >= 1) in the object

Usage

```
numAnchors(x)

## S4 method for signature 'loops'
numAnchors(x)
```

Arguments

`x`	A loops object to be summarized

Details

This function returns a data.frame where the column names specify the sample in the original `loops` object and the only row shows the number of anchors used to support that sample

Value

A data.frame of each sample and the number of anchors

Examples

```
# Show number of anchors each sample is supported by
rda<-paste(system.file('rda',package='diffloop'),'loops.small.rda',sep='/')
load(rda)
numAnchors(loops.small)
```

numLoops *Per-sample loop quantities*

Description

`numLoops` counts number of loops for each sample based on the index of nloops and returns a data.frame

Usage

```
numLoops(dlo, nloops = 1:10)

## S4 method for signature 'loops,numeric'
numLoops(dlo, nloops = 1:10)

## S4 method for signature 'loops,missing'
numLoops(dlo, nloops = 1:10)
```

Arguments

`dlo`	A loops object
`nloops`	A numeric vector of counts to be considered

Details

This function shows the number of unique loops with at least `nloops` in counts. Can be used to quickly visualize relative sequencing depth between samples

Value

A data.frame

Examples

```
# Determine what samples have loops with 1-20 counts
rda<-paste(system.file('rda',package='diffloop'),'loops.small.rda',sep='/')
load(rda)
nLoops <- numLoops(loops.small, 1:20)

# Determine what samples loops with 1-10 counts by default
nLoops <- numLoops(loops.small)
```

`padGRanges` *Pad a GRanges object*

Description

`padGRanges` takes a GRanges object and adds or substracts distance based on user-defined input. Upstream and downstream consider strand information when available. Specify only either pad or upstream/downstream when using

Usage

```
padGRanges(gro, upstream = 0, downstream = 0, pad = 0)

## S4 method for signature 'GRanges'
padGRanges(gro, upstream = 0, downstream = 0, pad = 0)
```

Arguments

`gro`	A granges object
`upstream`	Distance in BP added upstream
`downstream`	Distance in BP added downstream
`pad`	Distance in BP added

Value

A GRanges object with adjusted start and end values

Examples

```
#Read in CTCF Jurkat peaks in
ctcf_j <- system.file('extdata','Jurkat_CTCF_chr1.narrowPeak',package = 'diffloop')
ctcf <- bedToGRanges(ctcf_j)
ctcf.pad <- padGRanges(ctcf, pad = 1000)
```

`pcaPlot` *Visualize sample relationships*

Description

`pcaPlot` takes a loops object plots the individual samples based on the principal components of the loop counts matrix

Usage

```
pcaPlot(dlo)

## S4 method for signature 'loops'
pcaPlot(dlo)
```

Arguments

`dlo`	A loops object

Details

Groups for the principal component plots are derived from `colData` and the normalizing factors are also taken from `colData`. While some loops objects may have non-informative groups or size factors, they should always be present.

Value

A ggplot2 plot

Examples

```
rda<-paste(system.file('rda',package='diffloop'),'loops.small.rda',sep='/')
load(rda)
p1 <- pcaPlot(loops.small)
```

`plotTopLoops` *Plot the most significant loops*

Description

`plotTopLoops` takes a loops object and creates a time-stamped .pdf file with loop plots (one per page) of the top loops.

Usage

```
plotTopLoops(lto, n = 0, PValue = 1, FDR = 1, organism = "h",
  colorLoops = FALSE)

## S4 method for signature 'loops'
plotTopLoops(lto, n = 0, PValue = 1, FDR = 1,
  organism = "h", colorLoops = FALSE)
```

Arguments

`lto`	loops object
`n`	number of loops to print (can remain 0 to specify from other parameters) determined by PValue
`PValue`	Maximum pvalue threshold for loop inclusion when printing loop plot
`FDR`	False discovery rate threshold for inclusion
`organism`	Either 'm' for mouse or 'h' for human.
`colorLoops`	Default FALSE; specify true if rowData slot contains loop.type from annotateLoops to visualize plots with varying colors for CTCF looping and enhancer-promoter looping

Details

Each plot will show the region +/- 1 loopwidth of the loop with annotation specified for either human or mouse. Assumes columns Pvalue and FDR are specified in the loops object. We recommend removing self loops before using this function (and in reality, before any association testing was called.)

Value

Prints a time stamped .pdf file of top loops

Examples

```
rda<-paste(system.file('rda',package='diffloop'),'loops.small.rda',sep='/')
load(rda)
jpn.u <- removeSelfLoops(loops.small)
jpn_loopfit <- loopFit(jpn.u)
assoc_jn <- loopTest(jpn_loopfit, coef = 2)
plotTopLoops(assoc_jn, n=2)
```

---

quickAssoc *Perform quick differential loop calling*

Description

quickAssoc takes a loops object and performs a basic edgeR association on the counts matrix and groups from colData

Usage

```
quickAssoc(y)

## S4 method for signature 'loops'
quickAssoc(y)
```

Arguments

y	A loops object for association

Details

This function returns the output of fitting an edgeR model using the groups defined in colData for the specific loops object. The factor normalization is based on the edgeR model. For quick association, the number of groups is restricted to two. If a more complex group structure exists, consider using the loopFit and loopTest functions

Value

A loops object

Examples

```
# Differential loop calling between naive and primed
rda<-paste(system.file('rda',package='diffloop'),'loops.small.rda',sep='/')
load(rda)
np <- loops.small[,1:4]
assoc_np <- quickAssoc(np)
```

removeSelfLoops *Remove self loops*

Description

`removeSelfLoops` removes instances where a loop is observed between the same anchor

Usage

```
removeSelfLoops(dlo)

## S4 method for signature 'loops'
removeSelfLoops(dlo)
```

Arguments

`dlo`	A loops object

Details

This function removes loops from the `interactions` slot that reference the same index of the `anchors` slot.

Value

A loops object

Examples

```
rda<-paste(system.file('rda',package='diffloop'),'loops.small.rda',sep='/')
load(rda)
jpn_unique <- removeSelfLoops(loops.small)
```

rmchr *Remove 'chr' from GRanges seqnames*

Description

`rmchr` takes a loops object or GRanges object and simply removes the 'chr' from seqnames, if is present

Usage

```
rmchr(dlo)

## S4 method for signature 'loops'
rmchr(dlo)

## S4 method for signature 'GRanges'
rmchr(dlo)
```

Arguments

`dlo`	A loops object or GRanges object

Details

Often times, performing functions on GRanges objects can go awry if the seqnames are systematically different. A common example of this is when some GRanges objects has the format of 'chr1' while the other has '1'. We can remove 'chr' from the first object

Value

An identical loops/GRanges object except 'chr' removed

Examples

```
library(GenomicRanges)
regA <- GRanges(c('1'),IRanges(c(36200000),c(36300000)))
addchr(regA)
regA
rmchr(regA)
regA
```

---

sampleNames,loops-method

Grab/Update Sample Names

Description

`sampleNames` takes a loops object returns the names of the samples in the structure. One can also update the names using set replace.

Usage

```
## S4 method for signature 'loops'
sampleNames(object)

## S4 replacement method for signature 'loops,ANY'
sampleNames(object) <- value
```

Arguments

`object`	A loops object
`value`	New names when using set replace

Details

The examples show both accession and updating sample names.

Value

Vector of sample names

Examples

```
rda<-paste(system.file('rda',package='diffloop'),'loops.small.rda',sep='/')
load(rda)
sampleNames(loops.small)
nnames <- c('one', 'two', 'three', 'four', 'five', 'six')
sampleNames(loops.small) <- nnames
```

`slidingWindowTest` *Combined association test for all loops in a defined region*

Description

`slidingWindowTest` takes a `loops` object and integer values of the association window and the distance between consecutive windows.

Usage

```
slidingWindowTest(x, window, step)

## S4 method for signature 'loops,numeric,numeric'
slidingWindowTest(x, window, step)
```

Arguments

`x`	A loops object with PValue column (from association testing)
`window`	The length a window will be for combined association
`step`	The size that the window will shift for each association

Details

This function returns a data.frame sorted by FDR of each region. The engine loops over each chromosome and defines the first window at the left-most loop and slides the window right until no more loops are present in `x` Each region is determined from a sliding window of fixed length. The combined significance measure per feature is computed via the Simes method for intrachromosomal loops where at least one anchor from the loop overlaps with the region. Requires PValue column in the rowData slot.

Value

A data.frame sorted by FDR

Examples

```
# Sliding window test 100kb at a time between naive and jurkat
rda<-paste(system.file('rda',package='diffloop'),'loops.small.rda',sep='/')
load(rda)
jpn_loopfit <- loopFit(loops.small)
# Differential loop calling between naive and jurkat
assoc_jn <- loopTest(jpn_loopfit, coef = 2)
sw_jn <- slidingWindowTest(assoc_jn, 100000, 100000)
```

---

splitSamples *Split samples into their own loops object*

Description

`splitSamples` takes a loops object and returns a list of loops objects where each sample populates its own loops object

Usage

```
splitSamples(dlo)

## S4 method for signature 'loops'
splitSamples(dlo)
```

Arguments

`dlo`	A loops object

Details

This function splits the colData and counts slots for each sample but makes copies of the anchors, interactions, and rowdata

Value

A list of loops objects w

Examples

```
# Updating groups from all 'group1' to meaningful designations
rda<-paste(system.file('rda',package='diffloop'),'loops.small.rda',sep='/')
load(rda)
split <- splitSamples(loops.small)
```

`subsetLoops` *Subset loops*

Description

`subsetLoops` restricts the loops and counts matrix to only those specified by `idxa`, either numerically or logically

Usage

```
subsetLoops(dlo, idxa)

## S4 method for signature 'loops,logical'
subsetLoops(dlo, idxa)

## S4 method for signature 'loops,numeric'
subsetLoops(dlo, idxa)
```

Arguments

`dlo`	A loops object
`idxa`	A numeric vector or logical vector

Details

This function returns a loops object where the loops are retained only if they meet a logical criteria or are included in the numeric vector of `idxa`. Only the anchors that reference a loop in the subsetted loops object are retained.

Value

A loops object

Examples

```
# Return the first 10 loops
rda<-paste(system.file('rda',package='diffloop'),'loops.small.rda',sep='/')
load(rda)
#' ten <- subsetLoops(loops.small, 1:10)

# Subset loops with widths greater than 10000
big <- subsetLoops(loops.small, loopWidth(loops.small) >= 10000)
```

subsetRegion — *Extract region from loops object*

Description

subsetRegion takes a loops object and a GRanges object and returns a loops object where both anchors map inside the GRanges coordinates by default. Once can specify where only one anchor is in the region instead.

Usage

```
subsetRegion(dlo, region, nanchors)

## S4 method for signature 'loops,GRanges,numeric'
subsetRegion(dlo, region, nanchors)

## S4 method for signature 'loops,GRanges,missing'
subsetRegion(dlo, region, nanchors)
```

Arguments

dlo	A loops object to be subsetted
region	A GRanges object containing region of interest
nanchors	Number of anchors to be contained in GRanges object. Default 2

Details

By default, nachors = 2, meaning both anchors need to be in the region for the loop to be preserved when extracting. However, by specifying a numeric 1, interactions with either the left or right anchor will be extracted. Loops with both anchors in the region will be excluded (exclusive or). To get an inclusive or, take the union of subsetting both with 1 and 2.

Value

A loops object

Examples

```
# Grab region chr1:36000000-36100000
library(GenomicRanges)
regA <- GRanges(c('1'),IRanges(c(36000000),c(36100000)))
rda<-paste(system.file('rda',package='diffloop'),'loops.small.rda',sep='/')
load(rda)
# Both anchors in region
loops.small.two <- subsetRegion(loops.small, regA)
#Only one anchor in region
loops.small.one <- subsetRegion(loops.small, regA, 1)
#Either one or two anchors in region
loops.small.both <- union(loops.small.one, loops.small.two)
```

`summary,loops-method` *Link the anchors and interactions back together*

Description

`summary` takes a `loops` object and breaks the loop data structure resulting in a `data.frame`.

Usage

```
## S4 method for signature 'loops'
summary(object)
```

Arguments

`object`	A loops object to be summarized

Details

This function returns a `data.frame` where the left and right anchors are visualized together along with the loop width, individual counts, and any anchor meta-data that has been annotated into the anchors GRanges object as well as any rowData varianble

Value

A data.frame

Examples

```
# Summarizing the first ten loops in \code{loops.small}
rda<-paste(system.file('rda',package='diffloop'),'loops.small.rda',sep='/')
load(rda)
summarydf <- summary(loops.small[1:10,])
# Summarizing the loops and significance results between naive and primed
summarylt <- summary(quickAssoc(loops.small[,1:4])[1:10,])
```

`tail,loops-method` *Extract last part of loops object*

Description

Extract last part of loops object

Usage

```
## S4 method for signature 'loops'
tail(x, n = 6, ...)
```

Arguments

`x`	A loops object
`n`	Number of lines to view
`...`	Other non-essential params

Value

A loops object

`topLoops` *Grab top loops*

Description

`topLoops` takes a loops object and performs basic filtering for `FDR` or `PValue`

Usage

```
topLoops(dlo, FDR, PValue)

## S4 method for signature 'loops,numeric,numeric'
topLoops(dlo, FDR, PValue)

## S4 method for signature 'loops,numeric,missing'
topLoops(dlo, FDR, PValue)

## S4 method for signature 'loops,missing,numeric'
topLoops(dlo, FDR, PValue)
```

Arguments

`dlo`	A loops object
`FDR`	Maximum threshold for False Discovery Rate; default = 1
`PValue`	Maximum threshold for P-value; default = 1

Details

This function returns a subsetted `loops` object where all loops meet the significance threshold speci-ficed by the parameters in the function call

Value

A loops object subsetted by specified parameters

Examples

```
# Differential loop calling between naive and primed
rda<-paste(system.file('rda',package='diffloop'),'loops.small.rda',sep='/')
load(rda)
np <- loops.small[,1:4]
assoc_np <- quickAssoc(np)
top_np <- topLoops(assoc_np, FDR = 0.3)
```

`union,loops,loops-method` *Combine two loops objects*

Description

`union` combines two loops objects' interactions and anchors and populates the `colData` matrix where available

Usage

```
## S4 method for signature 'loops,loops'
union(x, y)
```

Arguments

`x`	A loops object
`y`	A loops object

Details

This function returns a single loops object that has all the anchors and interactions contained in the two loops objects that were part of the input. However, when the two objects have different samples, the counts matrix will contain missing values (e.g. when loop counts in x are not in y, those values are unknown). While the number of interactions, colData, and anchors should be correct, we need to correct the counts using a subsetting function. The row data gets re-initialized here to only the loop widths

Value

A loops obect

Examples

```
# divide and recombine samples
rda<-paste(system.file('rda',package='diffloop'),'loops.small.rda',sep='/')
load(rda)
naive <- loops.small[,1:2]
primed <- loops.small[,3:4]
np <- union(naive, primed)
```

```
# Subset from full to get correct counts
c.np <- loopsSubset(np, loops.small)
```

`updateLDGroups` *Update groups in colData for loops object*

Description

`updateLDGroups` changes the `groups` column in `colData` for a `loops` object

Usage

```
updateLDGroups(dlo, groups)

## S4 method for signature 'loops'
updateLDGroups(dlo, groups)
```

Arguments

`dlo`	A loops object
`groups`	A character vector. Lists the groups each sample belongs in

Details

This function updates the `groups` column in `colData` for a `loops` object. Make sure that the length of `groups` the number of samples in `colData`!

Value

A loops object with new groups in `colData`

Examples

```
# Updating groups from all 'group1' to meaningful designations
rda<-paste(system.file('rda',package='diffloop'),'loops.small.rda',sep='/')
load(rda)
celltypes <- c('naive1','naive1','primed2','primed2','jurkat3','jurkat3')
loops.small <- updateLDGroups(loops.small, celltypes)
```

`[,loops,numeric,numeric,missing-method`
Extract parts of a loops object

Description

Extract parts of a loops object

Usage

```
## S4 method for signature 'loops,numeric,numeric,missing'
x[i, j, drop]

## S4 method for signature 'loops,missing,numeric,missing'
x[i, j, drop]

## S4 method for signature 'loops,numeric,missing,missing'
x[i, j, drop]
```

Arguments

`x`	A loops object for subsetting
`i`	Loops to be subsetted
`j`	Samples to be subsetted
`drop`	Other non-essential parameters needed for sub

Value

A loops object

Index