

# Package ‘diffloop’

October 17, 2017

**Type** Package

**Title** Identifying differential DNA loops from chromatin topology data

**Version** 1.4.0

**Date** 2016-11-30

**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 or other assays that infer chromatin loops.

**Imports** methods, GenomicRanges, foreach, plyr, dplyr, reshape2, ggplot2, matrixStats, Sushi, edgeR, locfit, statmod, biomaRt, GenomeInfoDb, S4Vectors, IRanges, grDevices, graphics, stats, utils, Biobase, readr, data.table, rtracklayer, pbapply, limma

**License** MIT + file LICENSE

**URL** <https://github.com/aryeelab/diffloop>

**BugReports** <https://github.com/aryeelab/diffloop/issues>

**LazyData** TRUE

**Suggests** DESeq2, diffloopdata, ggrepel, 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' 'mangoFunctions.R'  
'plotting.R' 'ruan.R'

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

**NeedsCompilation** no

## R topics documented:

addchr . . . . .	3
annotateAnchors . . . . .	4
annotateAnchors.bed . . . . .	5

annotateAnchors.bigwig	6
annotateLoops	7
annotateLoops.dge	8
bedToGRanges	9
calcLDSizeFactors	9
callCCDs	10
computeBoundaryScores	11
diffloop	12
dim.loops-method	12
featureTest	13
filterLoops	13
filterSpanningLoops	14
geneinfo	15
getHumanGenes	16
getHumanTSS	16
getMouseGenes	17
getMouseTSS	18
head.loops-method	19
human.genes	19
interchromosomal	20
intrachromosomal	20
keepCTCFloops	21
keepEPloops	22
loopAssoc	23
loopDistancePlot	24
loopGenes	25
loopMetrics	25
loopPlot	26
loops-class	27
loops.small	28
loopsMake	28
loopsMake.mango	29
loopsSubset	30
loopWidth	31
mangoCorrection	32
manyLoopPlots	33
mergeAnchors	34
numAnchors	35
numLoops	35
padGRanges	36
pcaPlot	37
plotTopLoops	38
quickAssoc	39
quickAssocVoom	39
removeRegion	40
removeSelfLoops	41
rmchr	42
sampleNames.loops-method	42
slidingWindowTest	43
splitSamples	44
subsetLoops	45
subsetRegion	46

<i>addchr</i>	3
subsetRegionAB . . . . .	47
summary,loops-method . . . . .	47
tail,loops-method . . . . .	48
topLoops . . . . .	49
union,loops,loops-method . . . . .	50
updateLDGroups . . . . .	51
[,loops,numeric,numeric,missing-method . . . . .	51
<b>Index</b>	<b>53</b>

---

<i>addchr</i>	<i>Add 'chr' to GRanges seqnames</i>
---------------	--------------------------------------

---

### 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 GRanges*

---

### 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)
```

---

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

---

## Description

annotateAnchors.bed adds a numeric variable to meta data columns in the anchors slot based on a user-specified .bed file where the fourth column is a numeric score.

## Usage

```
annotateAnchors.bed(dlo, file, FUN = mean, pad = 0)
```

```
## S4 method for signature 'ANY'  
annotateAnchors.bed(dlo, file, FUN = mean, pad = 0)
```

## Arguments

dlo	A loops object whose anchors will be annotated
file	A string pointing to the bed file of interest
FUN	A function used to combine multiple values observed in a single anchor; default is mean
pad	An integer value of to pad the anchors of the loops object; default is 0

## Details

This function adds a meta data column to anchors of the specified loops object. All values from the .bed file that overlap with the each anchor are handled by the FUN (default is to average them) to produce a single value added to the mcols of the anchors.

## Value

A loops object with new numeric 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')
```

annotateAnchors.bigwig

*Add meta data column to anchors based on .bigwig*

---

## Description

annotateAnchors.bigwig adds a numeric variable to meta data columns in the anchors slot based on a user-specified .bigwig file.

## Usage

```
annotateAnchors.bigwig(dlo, file, FUN = mean, pad = 0)
```

```
## S4 method for signature 'ANY'
```

```
annotateAnchors.bigwig(dlo, file, FUN = mean, pad = 0)
```

## Arguments

dlo	A loops object whose anchors will be annotated
file	A file corresponding to the bigwig of interest
FUN	A function used to combine multiple values observed in a single anchor; default is mean
pad	An integer value of to pad the anchors of the loops object; default is 0

## Details

This function adds a meta data column to anchors of the specified loops object. All values from the .bigwig file that overlap with the each anchor are handled by the FUN (default is to average them) to produce a single value added to the mcols of the anchors.

## Value

A loops object with new numeric 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')
```

---

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), p-p (promoter-promoter), e-e (enhancer-enhancer), ctcf (CTCF-CTCF) or none (no biological annotation).

### Usage

```

annotateLoops(lto, ctcf, enhancer, promoter)

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

## S4 method for signature 'loops,missing,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)
# annotated_small <- annotateLoops(loops.small, ctcf, h3k27ac, promoter)

```

---

annotateLoops.dge      *Annotate enhancer-promoter loops with differential gene expression*

---

## Description

annotateLoops.dge adds columns to the rowData slot of a loops object that shows summary statistics corresponding TSS of a gene name based on the genes.tss rowData column. This function should be used following the keepEPloops function.

## Usage

```
annotateLoops.dge(lto, deseq_res, multiple = FALSE)

## S4 method for signature 'ANY'
annotateLoops.dge(lto, deseq_res, multiple = FALSE)
```

## Arguments

lto	A loops object whose loops will be annotated
deseq_res	A data.frame
multiple	Annotate loops with multiple TSS? Default = FALSE

## Details

This function links enhancer-promoter loops and differential gene expression results. The rownames of the deseq\_res slots should correspond to the gene names in the gene.tss column of the rowData slot of the loops object. The function returns a loops object if multiple is specified as FALSE which is the case by default. Otherwise, if multiple is TRUE, then this function returns a data frame since each loop may have more than one TSS. One can reproduce this dataframe quickly when multiple = FALSE using the summary() function on the returned loops object.

## Value

A loops object if multiple = FALSE or data frame if multiple = TRUE

## Examples

```
rda<-paste(system.file('rda',package='diffloop'),'loops.small.rda',sep='/')
load(rda)
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)
#small.ep <- keepEPloops(loops.small, h3k27ac, promoter)
#ADD SOMETHING HERE.
```



---

bedToGRanges	<i>Read a file and make a GRanges object</i>
--------------	--

---

**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 specifying .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	<i>Compute normalizing factors for each sample</i>
-------------------	--

---

**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)
```

---

callCCDs

*Compute Chromatin Contact Domains (CCDs)*

---

**Description**

callCCDs determines regions

**Usage**

```
callCCDs(lo, petWeights = FALSE, lowCoveragePercentile = 0.05)
```

```
## S4 method for signature 'ANY'
callCCDs(lo, petWeights = FALSE,
  lowCoveragePercentile = 0.05)
```

**Arguments**

lo                    A loops object

petWeights          Boolean to weight loop coverage by number of PETs. Default = FALSE

lowCoveragePercentile  
                     Percentile of low coverage to be dropped. Default = 0.05

**Details**

This function returns a GRanges object of regions determined to be Chromatin Contact Domains as defined in the Tang et al. 2015 paper from the Ruan group. Users can choose to weight the loops by the total number PETs (across all samples) or not and what percent. For details of this method, see page 12 of the supplement of PMID:26686651. Make sure there are only loops within a chromosome before calling this.

**Value**

A GRanges object of called Chromatin Contact domains

**Examples**

```
rda<-paste(system.file('rda',package='diffloop'),'loops.small.rda',sep='/')
load(rda)
#lo <- subsetLoops(loops.small, c(1,2,5,6,7,8,9,27,69))
#ccd <- callCCDs(lo, petWeights = TRUE, lowCoveragePercentile = 0.5)
```

---

computeBoundaryScores *Compute boundary scores for genomic loci in between anchors*

---

**Description**

computeBoundaryScores determines the boundary scores corresponding to the Genomic region either in between pairs of anchors. To achieve this, the number of PETs for a set of samples (default is all = 0) is summed over a window (default 1MB) on the left (A) and the right (B) of gap. Thus sum of the number of PETs in these windows is divided by the number of PETs that span the two loci, plus 1 (C). A larger value corresponds to a stronger boundary.

**Usage**

```
computeBoundaryScores(dlo, samples = 0, windowSize = 5e+05)

## S4 method for signature 'loops'
computeBoundaryScores(dlo, samples = 0,
  windowSize = 5e+05)
```

**Arguments**

dlo	A loops object
samples	= 0 Vector indexing which samples should be used. 0 is all
windowSize	= 500000 BP length on left and right of putative boundary to define A/B

**Details**

Warning: this function is slow; there is a progress bar outputted to give an anticipated runtime.

**Value**

A GRanges object with genomic loci and boundary scores in the mcols

**Examples**

```
# Return the width for loops
rda<-paste(system.file('rda',package='diffloop'),'loops.small.rda',sep='/')
load(rda)
# BS <- computeBoundaryScores(loops.small, samples = 0, windowSize = 500000)
```

---

diffloop	<i>diffloop: A package for differential DNA loop calling from ChIA-PET data</i>
----------	---

---

### 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	<i>See dimensions of loops object</i>
------------------	---------------------------------------

---

### 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	<i>Combined association test for all loops in a defined region</i>
-------------	--

---

### 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)
# assoc <- loopAssoc(loops.small, coef = 2)
# Gene based association
# sw_jn <- featureTest(assoc, getHumanGenes(c('1')))
```

---

filterLoops	<i>Filter loops</i>
-------------	---------------------

---

### 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.jp1 <- filterLoops(loops.small, 5000, 0, 0)
# Restrict loops to > 5kb width and have >= 3 replicates in >= 1 sample
filtered.jp2 <- filterLoops(loops.small, 5000, 3, 1)
# Restrict loops to > 10kb width and have >= 3 replicates in >= 2 samples
filtered.jp3 <- filterLoops(loops.small, 10000, 3, 2)
```

---

filterSpanningLoops     *Retain loops spanning some genomic feature*

---

**Description**

filterSpanningLoops returns a loops object where the ends of the anchors completely span one or more genomic feature (e.g. boundary)

**Usage**

```
filterSpanningLoops(dlo, gf)

## S4 method for signature 'loops,GRanges'
filterSpanningLoops(dlo, gf)
```

**Arguments**

**dlo** A loops object  
**gf** A GRanges object of features

**Details**

Rather than a simple overlap, the function by default requires a genomic locus to be completely contained

**Value**

An loops object

**Examples**

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

---

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 or h

**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	<i>Get protein coding gene regions</i>
---------------	--

---

### Description

getHumanGenes returns a GRanges object of all protein coding genes genome-wide or within specified chromosomes. Annotation is from regions from hg19/Gr37 and protein coding genes.

### 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) or specified 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	<i>Get Human Transcription Start Sites</i>
-------------	--

---

### Description

getHumanTSS returns a GRanges object of all transcription start sites for humans. Regions from hg19/Gr37 for protein coding regions.



**Usage**

```
getHumanTSS(chr)

## S4 method for signature 'missing'
getHumanTSS(chr)

## S4 method for signature 'character'
getHumanTSS(chr)
```

**Arguments**

chr                    Specifies what chromosomes are desired for the TSS#'

**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()
```

---

getMouseGenes	<i>Get protein coding gene regions</i>
---------------	--

---

**Description**

getMouseGenes returns a GRanges object of all protein coding genes genome-wide or within specified chromosomes. Annotation is from regions from mm9 and protein coding genes.

**Usage**

```
getMouseGenes(chr)

## S4 method for signature 'missing'
getMouseGenes(chr)

## S4 method for signature 'character'
getMouseGenes(chr)
```

**Arguments**

chr                    A vector of chromosomes

**Details**

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

**Value**

A GRanges object

**Examples**

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

---

getMouseTSS

*Get Mouse Transcription Start Sites*

---

**Description**

getMouseTSS returns a GRanges object of all transcription start sites for humans. Regions from mm9 for protein coding regions.

**Usage**

```
getMouseTSS(chr)

## S4 method for signature 'missing'
getMouseTSS(chr)

## S4 method for signature 'character'
getMouseTSS(chr)
```

**Arguments**

chr                    Specifies what chromosomes are desired for the TSS#

**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
mouse.TSS <- getMouseTSS()
```

---

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

---

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 an 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)
```

---

keepCTCFloops	<i>Keep CTCF loops</i>
---------------	------------------------

---

**Description**

keepCTCFloops returns loops that are nearby CTCF peaks as determined by some external data in a GRanges object

**Usage**

```
keepCTCFloops(lto, ctcf)

## S4 method for signature 'loops,GRanges'
keepCTCFloops(lto, ctcf)
```

**Arguments**

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

**Details**

This function works similar to the `annotateLoops` function but returns only CTCF loops that are defined in this function. However, loops in `annotateLoops` may have a different annotation based on their priority scheme. For example, an e-p loop from `annotateLoops` may be returned as a CTCF loop by this function if the loop had both annotations. These peaks don't necessarily need to be CTCF peaks, so using a `GRanges` object with enhancers or promoters to determine e-e loops and p-p loops could also be used in this function

**Value**

A loops object with all loops having both anchors in the `GRanges` region

**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)
# small.ctcf <- keepCTCFloops(loops.small, ctcf)
```

---

keepEPloops

*Keep enhancer-promoter loops*

---

**Description**

`keepEPloops` adds a column to the `rowData` slot of a loops object that shows the corresponding TSS of a gene name based on the promoter `GRanges`. The loops object is then subsetted and returns only loops that are enhancer-promoter.

**Usage**

```
keepEPloops(lto, enhancer, promoter)

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

**Arguments**

<code>lto</code>	A loops object whose loops will be annotated
<code>enhancer</code>	<code>GRanges</code> object corresponding to locations of enhancer peaks
<code>promoter</code>	<code>GRanges</code> object corresponding to locations of promoter regions

**Details**

This function works similar to the `annotateLoops` function but returns only enhancer-promoter loops that are defined in this function. Additionally, this function returns the gene name(s) of the nearby transcription start sites in a comma-separated list if there are multiple. These gene names are defined by the promoter `GRanges` `mcol` slot.

**Value**

A loops object with an additional row 'loop.type' in the rowData slot in addition to the gene.tss (which has the gene name) and the anchor.tss which shows the anchor(s) near the promoter region for the gene.

**Examples**

```
rda<-paste(system.file('rda',package='diffloop'),'loops.small.rda',sep='/')
load(rda)
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)
# small.ep <- keepEPloops(loops.small, h3k27ac, promoter)
```

---

loopAssoc

*Generalized differential Loop Calling*


---

**Description**

loopAssoc takes a loops object and prepares it for the returns another loops object with summary statistics per-loop in the rowData

**Usage**

```
loopAssoc(y, method = "edgeR", design = NULL, coef = NULL,
          contrast = NULL)
```

```
## S4 method for signature 'loops'
loopAssoc(y, method = "edgeR", design = NULL,
          coef = NULL, contrast = NULL)
```

**Arguments**

y	A loops object for association
method	Specifies association; either "Voom" or "edgeR"
design	A design matrix of the samples; required for "Voom"
coef	A vector for the coefficient of GLM. See edgeR manual
contrast	A vector for the contrast. See edgeR manual

**Details**

By the default, we generate is to generate a design matrix from loops@colData\$groups. Currently, 'edgeR' and 'Voom' are the two supported association methods, but new association tests may be added in later developments.

**Value**

A loops object

## Examples

```
# Differential loop fit
rda<-paste(system.file('rda',package='diffloop'),'loops.small.rda',sep='/')
load(rda)
# assoc <- loopAssoc(loops.small, coef = 2)
```

---

loopDistancePlot	<i>Visualize proportion of loops at distances</i>
------------------	---

---

## Description

loopDistancePlot takes a loops object plots the individual samples based on the proportion of PET counts at various distances

## Usage

```
loopDistancePlot(dlo)

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

## Arguments

dlo                    A loops object

## Details

Distances for the loops are taken from the rowData slot.

## Value

A ggplot2 plot

## Examples

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



---

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)

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

**Arguments**

dlo                    A loops object  
genesGR                A GRanges object of genes in first mcol.

**Details**

Function that annotates all loops. If there are multiple, the function returns a comma separated list. Adds a "loopGenes" column to the rowData slot. If the genesGR is left blank, diffloop will use protein coding genes for human from hg19.

**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 object and shows all loops in region (where both anchors are present)

**Usage**

```
loopPlot(x, y, organism = "h", geneinfo = "NA", colorLoops = TRUE,
         cache = TRUE, showAnchorWidths = FALSE, maxCounts = -1)

## S4 method for signature 'loops,GRanges'
loopPlot(x, y, organism = "h", geneinfo = "NA",
         colorLoops = TRUE, cache = TRUE, showAnchorWidths = FALSE,
         maxCounts = -1)
```

**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
showAnchorWidths	Display the width of the anchor on the plot? Default = FALSE
maxCounts	Number of counts associated with thickest loop. Default is largest count in region displayed

**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; orange are promoter-promoter loops; and purple are enhancer-enhancer loops. Plots use hg19 and mm9 annotation by default.

**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 <- loopPlot(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 (columns) for each sample (rows)  
rowData A data.frame with features (columns) for each loop (rows)

---

loops.small	<i>chr1:36000000-36300000 loops</i>
-------------	-------------------------------------

---

**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	<i>Read preprocessed ChIA-PET data from dnalooop</i>
-----------	--

---

**Description**

loopsMake reads in a data directory created by the dnalooop 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	Specificities 'intra', 'inter', or 'all' looping. Default 'all'

**Details**

This function reads in preprocessed ChIA-PET data produced by the dnaLoop preprocessing pipeline. The samples argument specifies which samples are read. If samples is not specified all samples will be read. The type option restricts loops whether they are on the same 'intra' or different 'inter' chromosome. Default is 'all'.

IMPORTANT: Assumes the delimiter is a space, not a tab on the files.

**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, 'inter')
```

---

loopsMake.mango

*Read preprocessed ChIA-PET data from mango*

---

**Description**

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

**Usage**

```
loopsMake.mango(beddir, samples = NA, mergegap = 500, ext = "all")

## S4 method for signature 'ANY'
loopsMake.mango(beddir, samples = NA, mergegap = 500,
  ext = "all")
```

**Arguments**

beddir	A string. The preprocessed data directory; Required
samples	A character vector. Optional list of samples to read in
mergegap	An integer value of the radius to merge anchors; default 500
ext	Specificities 'all' or 'fdr' file format; default 'all'

**Details**

This function reads in preprocessed ChIA-PET data produced by the mango preprocessing pipeline. The `samples` argument specifies which samples are read. If `samples` is not specified all samples will be read. The `ext` specifies which type of file to look for, either `all` or `fdr`, with `all` being the default. Under the default, all samples with the extension `.fdr.mango` will be processed. Finally, the FDR parameter (default = 1) specifies the minimum threshold for loops to be added to the greater loops object. Currently, we do not support importing the verbose output, so the `verbose` parameter when executing mango should be set to `FALSE` or the user will have to parse the file before reading into `diffloop` using `awk`, `cut`, or something similar.

**Value**

A loops object where 'chr' is removed from the anchors.

**Examples**

```
# UPDATE THIS
bd<- system.file('extdata', 'esc_jurkat', package='diffloopdata')
```

---

loopsSubset	<i>Subset two difloop objects</i>
-------------	-----------------------------------

---

**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 object

**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)
```

---

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 positive 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)
```

---

mangoCorrection	<i>Perform mango bias correction</i>
-----------------	--------------------------------------

---

## Description

mangoCorrection takes a loops object and filters loops based on the binomial model used in the mango ChIA-PET pipeline.

## Usage

```
mangoCorrection(lo, FDR = 1, PValue = 1, nbins = 10)
```

```
## S4 method for signature 'ANY'
```

```
mangoCorrection(lo, FDR = 1, PValue = 1, nbins = 10)
```

## Arguments

lo	A loops object.
FDR	Minimum FDR value for loop to be included; default 1
PValue	Minimum p0value for loop to be included; default 1
nbins	Number of bins for mango computation

## Details

This function processes ChIA-PET data in a loops object and filters loops that may be biased due to proximity or low PET counts as previously described by the mango pipeline. PET and anchor counts are aggregated across all samples to compute statistical significance. Consider using a larger number of bins (e.g. 30) for a larger data object when possible.

## Value

A loops object where loops are filtered using mango bias correction

## Examples

```
rda <- paste(system.file("rda", package = "diffloop"), "loops.small.rda", sep = "/")
load(rda)
loops.small <- removeSelfLoops(loops.small)
loops.small.mango <- mangoCorrection(loops.small, PValue = 0.05)
```



---

manyLoopPlots	<i>Plot several loop regions</i>
---------------	----------------------------------

---

### Description

manyLoopPlots takes a loops object and creates a time-stamped .pdf file with loop plots (one per page) of all regions specified in the GRanges object.

### Usage

```
manyLoopPlots(x, y, organism = "h", geneinfo = "NA", colorLoops = FALSE,
  cache = TRUE, maxCounts = -1, showAnchorWidths = FALSE)
```

```
## S4 method for signature 'loops,GRanges'
manyLoopPlots(x, y, organism = "h",
  geneinfo = "NA", colorLoops = FALSE, cache = TRUE, maxCounts = -1,
  showAnchorWidths = FALSE)
```

### Arguments

x	loops object
y	GRanges object with many regions to be visualized
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
maxCounts	Number of counts associated with thickest loop. Default is largest count in region displayed
showAnchorWidths	Display the width of the anchor on the plot? Default = FALSE

### Details

Each plot will show one region sequentially that is supplied in the GRanges object.

### 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)
assoc <- loopAssoc(jpn.u, coef = 2)
#manyLoopPlots(assoc, regs) #define regs as multiple GRanges
```

---

mergeAnchors	<i>Combine nearby anchors into one peak</i>
--------------	---

---

## 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	<i>Get number of anchors in each sample</i>
------------	---

---

**Description**

numAnchors takes a loops object and 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	<i>Per-sample loop quantities</i>
----------	-----------------------------------

---

**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 subtracts 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)
assoc <- loopAssoc(jpn.u,coef = 2)
plotTopLoops(assoc, n=2)
```

---

quickAssoc	<i>Perform quick differential loop calling</i>
------------	--

---

**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 loopAssoc function.

**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)
```

---

quickAssocVoom	<i>Perform quick differential loop calling</i>
----------------	--

---

**Description**

quickAssocVoom takes a loops object and performs a basic voom association on the counts matrix and groups from colData

**Usage**

```
quickAssocVoom(y)

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

**Arguments**

y                    A loops object for association

**Details**

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

**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_voom <- quickAssocVoom(np)
```

---

removeRegion	<i>Remove region from loops object</i>
--------------	--

---

**Description**

removeRegion takes a loops object and a GRanges object and returns a loops object where neither anchors map inside the GRanges coordinates.

**Usage**

```
removeRegion(dlo, region)

## S4 method for signature 'loops,GRanges'
removeRegion(dlo, region)
```

**Arguments**

dlo                    A loops object to be subsetted  
region                  A GRanges object containing region of interest

**Value**

A loops object with no anchors touching the region given



**Examples**

```
# Remove 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)
# Get rid of loop if either anchor touches that region
restricted <- removeRegion(loops.small, regA)
```

---

removeSelfLoops	<i>Remove self loops</i>
-----------------	--------------------------

---

**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	<i>Remove 'chr' from GRanges seqnames</i>
-------	---

---

### 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	<i>Grab/Update Sample Names</i>
--------------------------	---------------------------------

---

### 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	<i>Combined association test for all loops in a defined region</i>
-------------------	--

---

**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)
# assoc_jn <- loopAssoc(loops.small, 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 with one sample per index.

**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	<i>Extract region from loops object</i>
--------------	---

---

### 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, nanchors = 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)
```

---

subsetRegionAB	<i>Retain loops that have anchors in two specified regions</i>
----------------	--

---

**Description**

subsetRegionAB returns a loops object where one anchor maps to regionA and the other maps to region B

**Usage**

```
subsetRegionAB(dlo, regionA, regionB)

## S4 method for signature 'loops,GRanges,GRanges'
subsetRegionAB(dlo, regionA, regionB)
```

**Arguments**

dlo	A loops object
regionA	A GRanges object
regionB	A GRanges object

**Value**

A loops object

**Examples**

```
# Return the width for loops
library(GenomicRanges)
rda<-paste(system.file('rda',package='diffloop'),'loops.small.rda',sep='/')
load(rda)
regA <- GRanges(c('1'),IRanges(c(36000000),c(36100000)))
regB <- GRanges(c('1'),IRanges(c(36200000),c(36300000)))
splits <- subsetRegionAB(loops.small, regA, regB)
```

---

summary,loops-method	<i>Link the anchors and interactions back together</i>
----------------------	--

---

**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` variable. Finally, the `region` column contains the coordinates that readily facilitates visualization of loop in UCSC or DNALandscapeR by padding the loop by 25kb on either side.

**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	<i>Extract last part of loops object</i>
-------------------	--

---

**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	<i>Grab top loops</i>
----------	-----------------------

---

### 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 subsetting loops object where all loops meet the significance threshold specified by the parameters in the function call.

### Value

A loops object subsetting 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 object

## 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	<i>Update groups in colData for loops object</i>
----------------	--

---

**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	<i>Extract parts of a loops object</i>
--	--

---

**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

- \*Topic **datasets**
  - geneinfo, [15](#)
  - human.genes, [19](#)
  - loops.small, [28](#)
- [, loops, missing, numeric, missing-method
  - ([, loops, numeric, numeric, missing-method), [51](#)
- [, loops, numeric, missing, missing-method
  - ([, loops, numeric, numeric, missing-method), [51](#)
- [, loops, numeric, numeric, missing-method, [51](#)
- addchr, [3](#)
- addchr, GRanges-method (addchr), [3](#)
- addchr, loops-method (addchr), [3](#)
- annotateAnchors, [4](#)
- annotateAnchors, loops, GRanges, character, missing-method (annotateAnchors), [4](#)
- annotateAnchors, loops, GRanges, character, numeric-method (annotateAnchors), [4](#)
- annotateAnchors.bed, [5](#)
- annotateAnchors.bed, ANY-method (annotateAnchors.bed), [5](#)
- annotateAnchors.bigwig, [6](#)
- annotateAnchors.bigwig, ANY-method (annotateAnchors.bigwig), [6](#)
- annotateLoops, [7](#)
- annotateLoops, loops, GRanges, GRanges, GRanges-method (annotateLoops), [7](#)
- annotateLoops, loops, missing, GRanges, GRanges-method (annotateLoops), [7](#)
- annotateLoops.dge, [8](#)
- annotateLoops.dge, ANY-method (annotateLoops.dge), [8](#)
- bedToGRanges, [9](#)
- bedToGRanges, character-method (bedToGRanges), [9](#)
- calcLDSizeFactors, [9](#)
- calcLDSizeFactors, loops-method (calcLDSizeFactors), [9](#)
- callCCDs, [10](#)
- callCCDs, ANY-method (callCCDs), [10](#)
- computeBoundaryScores, [11](#)
- computeBoundaryScores, loops-method (computeBoundaryScores), [11](#)
- diffloop, [12](#)
- diffloop-package (diffloop), [12](#)
- dim, loops-method, [12](#)
- featureTest, [13](#)
- featureTest, loops, GRanges-method (featureTest), [13](#)
- filterLoops, [13](#)
- filterLoops, ANY-method (filterLoops), [13](#)
- filterSpanningLoops, [14](#)
- filterSpanningLoops, loops, GRanges-method (filterSpanningLoops), [14](#)
- geneinfo, [15](#)
- getHumanGenes, [16](#)
- getHumanGenes, character-method (getHumanGenes), [16](#)
- getHumanGenes, missing-method (getHumanGenes), [16](#)
- getHumanTSS, [16](#)
- getHumanTSS, character-method (getHumanTSS), [16](#)
- getHumanTSS, missing-method (getHumanTSS), [16](#)
- getMouseGenes, [17](#)
- getMouseGenes, character-method (getMouseGenes), [17](#)
- getMouseGenes, missing-method (getMouseGenes), [17](#)
- getMouseTSS, [18](#)
- getMouseTSS, character-method (getMouseTSS), [18](#)
- getMouseTSS, missing-method (getMouseTSS), [18](#)
- head, loops-method, [19](#)
- human.genes, [19](#)
- interchromosomal, [20](#)

- interchromosomal, loops-method (interchromosomal), 20
- intrachromosomal, 20
- intrachromosomal, loops-method (intrachromosomal), 20
- keepCTCFloops, 21
- keepCTCFloops, loops, GRanges-method (keepCTCFloops), 21
- keepEPloops, 22
- keepEPloops, loops, GRanges, GRanges-method (keepEPloops), 22
- loopAssoc, 23
- loopAssoc, loops-method (loopAssoc), 23
- loopDistancePlot, 24
- loopDistancePlot, loops-method (loopDistancePlot), 24
- loopGenes, 25
- loopGenes, loops, GRanges-method (loopGenes), 25
- loopGenes, loops, missing-method (loopGenes), 25
- loopMetrics, 25
- loopMetrics, loops-method (loopMetrics), 25
- loopPlot, 26
- loopPlot, loops, GRanges-method (loopPlot), 26
- loops (loops-class), 27
- loops-class, 27
- loops.small, 28
- loopsMake, 28
- loopsMake, ANY-method (loopsMake), 28
- loopsMake.mango, 29
- loopsMake.mango, ANY-method (loopsMake.mango), 29
- loopsSubset, 30
- loopsSubset, loops, loops-method (loopsSubset), 30
- loopWidth, 31
- loopWidth, loops-method (loopWidth), 31
- mangoCorrection, 32
- mangoCorrection, ANY-method (mangoCorrection), 32
- manyLoopPlots, 33
- manyLoopPlots, loops, GRanges-method (manyLoopPlots), 33
- mergeAnchors, 34
- mergeAnchors, loops, numeric, logical-method (mergeAnchors), 34
- mergeAnchors, loops, numeric, missing-method (mergeAnchors), 34
- numAnchors, 35
- numAnchors, loops-method (numAnchors), 35
- numLoops, 35
- numLoops, loops, missing-method (numLoops), 35
- numLoops, loops, numeric-method (numLoops), 35
- padGRanges, 36
- padGRanges, GRanges-method (padGRanges), 36
- pcaPlot, 37
- pcaPlot, loops-method (pcaPlot), 37
- plotTopLoops, 38
- plotTopLoops, loops-method (plotTopLoops), 38
- quickAssoc, 39
- quickAssoc, loops-method (quickAssoc), 39
- quickAssocVoom, 39
- quickAssocVoom, loops-method (quickAssocVoom), 39
- removeRegion, 40
- removeRegion, loops, GRanges-method (removeRegion), 40
- removeSelfLoops, 41
- removeSelfLoops, loops-method (removeSelfLoops), 41
- rmchr, 42
- rmchr, GRanges-method (rmchr), 42
- rmchr, loops-method (rmchr), 42
- sampleNames, loops-method, 42
- sampleNames<-, loops, ANY-method (sampleNames, loops-method), 42
- slidingWindowTest, 43
- slidingWindowTest, loops, numeric, numeric-method (slidingWindowTest), 43
- splitSamples, 44
- splitSamples, loops-method (splitSamples), 44
- subsetLoops, 45
- subsetLoops, loops, logical-method (subsetLoops), 45
- subsetLoops, loops, numeric-method (subsetLoops), 45
- subsetRegion, 46
- subsetRegion, loops, GRanges, missing-method (subsetRegion), 46

subsetRegion, loops, GRanges, numeric-method  
    (subsetRegion), 46  
subsetRegionAB, 47  
subsetRegionAB, loops, GRanges, GRanges-method  
    (subsetRegionAB), 47  
summary, loops-method, 47  
  
tail, loops-method, 48  
topLoops, 49  
topLoops, loops, missing, numeric-method  
    (topLoops), 49  
topLoops, loops, numeric, missing-method  
    (topLoops), 49  
topLoops, loops, numeric, numeric-method  
    (topLoops), 49  
  
union, loops, loops-method, 50  
updateLDGroups, 51  
updateLDGroups, loops-method  
    (updateLDGroups), 51