Package 'linkSet'

November 14, 2025

Title Base Classes for Storing Genomic Link Data

Version 1.0.0

Description Provides a comprehensive framework for representing, analyzing, and visualizing genomic interactions, particularly focusing on gene-enhancer relationships. The package extends the GenomicRanges infrastructure to handle paired genomic regions with specialized methods for chromatin interaction data from Hi-C, Promoter Capture Hi-C (PCHi-C), and single-cell ATAC-seq experiments. Key features include conversion from common interaction formats, annotation of promoters and enhancers, distance-based analyses, interaction strength metrics, statistical modeling using CHiCANE methodology, and tailored visualization tools. The package aims to standardize the representation of genomic interaction data while providing domain-specific functions not available in general genomic interaction packages.

License MIT + file LICENSE

Encoding UTF-8

Roxygen list(markdown = TRUE)

RoxygenNote 7.3.2

Depends GenomicRanges, S4Vectors, R (>= 4.5.0)

Imports methods, IRanges, GenomeInfoDb, BiocGenerics, Organism.dplyr, InteractionSet, ggplot2, patchwork, scales, foreach, iterators, stats, rlang, MASS, data.table, DBI, doParallel, AnnotationDbi

Suggests knitr, rmarkdown, testthat, TxDb.Mmusculus.UCSC.mm10.knownGene, TxDb.Hsapiens.UCSC.hg38.knownGene, TxDb.Hsapiens.UCSC.hg19.knownGene, org.Mm.eg.db, org.Hs.eg.db, GenomicFeatures, GenomicInteractions, gamlss, gamlss.tr, BiocStyle, rtracklayer

Collate class.R AllGenerics.R getset.R methods.R annotate.R distance.R formatConverter.R linkSet-package.R GRange_method.R test_helper.R visualization.R count.R statical.R data.R

biocViews Software, HiC, DataRepresentation, Sequencing, SingleCell, Coverage

URL https://github.com/GilbertHan1011/linkSet, https://gilberthan1011.github.io/linkSet

BugReports https://github.com/GilbertHan1011/linkSet/issues/new

VignetteBuilder knitr

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

git_branch RELEASE_3_22

git_last_commit e983433

git_last_commit_date 2025-10-29

Repository Bioconductor 3.22

Date/Publication 2025-11-13

Author Gilbert Han [aut, cre] (ORCID: <https://orcid.org/0009-0000-0293-2467>)

Maintainer Gilbert Han <GilbertHan1011@gmail.com>

Contents

`linkSet-package` *linkSet: Base Classes for Storing Genomic Link Data*

Description

Provides a comprehensive framework for representing, analyzing, and visualizing genomic interactions, particularly focusing on gene-enhancer relationships. The package extends the GenomicRanges infrastructure to handle paired genomic regions with specialized methods for chromatin interaction data from Hi-C, Promoter Capture Hi-C (PCHi-C), and single-cell ATAC-seq experiments. Key features include conversion from common interaction formats, annotation of promoters and enhancers, distance-based analyses, interaction strength metrics, statistical modeling using CHiCANE methodology, and tailored visualization tools. The package aims to standardize the representation of genomic interaction data while providing domain-specific functions not available in general genomic interaction packages.

Author(s)

Maintainer: Gilbert Han <GilbertHan1011@gmail.com> (ORCID)

See Also

Useful links:

- https://github.com/GilbertHan1011/linkSet
- https://gilberthan1011.github.io/linkSet
- Report bugs at https://github.com/GilbertHan1011/linkSet/issues/new

---

`annotateInter,linkSet-method`
Annotate linkSet with inter/intra chromosome interactions

Description

Annotate linkSet with inter/intra chromosome interactions

Usage

```
## S4 method for signature 'linkSet'
annotateInter(x)
```

Arguments

x	A linkSet object

Value

A linkSet object with an additional metadata column 'inter_type'

Examples

```
data(linkExample)
linkExample <- annotateInter(linkExample)
```

`annotatePromoter,linkSet-method`
Annotate the link set with txDb. Give a gene list, and return a

---

Description

Annotate the link set with txDb. Give a gene list, and return a

Usage

```
## S4 method for signature 'linkSet'
annotatePromoter(
  x,
  genome = "hg38",
  keyType = "symbol",
  upstream = 5000,
  overwrite = FALSE
)
```

Arguments

x	linkSet
genome	the genome you want to annotate
keyType	the key type. You can check with AnnotationDbi::keytypes
upstream	The upstream base from the gene
overwrite	Whether to overwrite the regionsBait if it already exists

Value

linkSet object

Examples

```
gr1 <- GRanges(seqnames = c("chr1", "chr2", "chr3"),
               ranges = IRanges(start = c(1000, 2000, 3000), width = 100),
               strand = "+", symbol = c("BRCA1", "TP53", "NONEXISTENT"))
gr2 <- GRanges(seqnames = c("chr1", "chr2", "chr3"),
               ranges = IRanges(start = c(5000, 6000, 7000), width = 100),
               strand = "+")
linkset_obj <- linkSet(gr1, gr2, specificCol = "symbol")

# Test annotatePromoter
annotated_linkset <- suppressWarnings(annotatePromoter(linkset_obj,
                                                       genome = "hg38",
                                                       upstream = 500,
                                                       overwrite = TRUE))
```

---

as.data.frame,linkSet-method

coerce linkSet to DataFrame

Description

coerce linkSet to DataFrame

Usage

```
## S4 method for signature 'linkSet'
as.data.frame(x)
```

Arguments

x	A linkSet object

Value

A DataFrame object

Examples

```
# Create a linkSet object
data(linkExample)
# Convert linkSet to DataFrame
df <- as.data.frame(linkExample)
print(df)
```

as.GInteractions *Convert to GInteractions*

Description

Convert linkSet object to GInteractions

Usage

```
as.GInteractions(x)

## S4 method for signature 'linkSet'
as.GInteractions(x)
```

Arguments

x	A linkset object

Value

A GInteractions object

Examples

```
data(linkExample)
gi <- as.GInteractions(linkExample)
gi
```

bait<- *linkSet-accessors*

Description

Methods to get and set fields in an linkSet object.

This method returns the anchor IDs of a linkSet object.

This method returns the anchors of a linkSet object.

This method returns the bait anchors of a linkSet object.

This method returns the other end (oe) anchors of a linkSet object.

This method is an alias for 'first' and returns the bait anchors of a linkSet object.

This method is an alias for 'second' and returns the other end (oe) anchors of a linkSet object.

This method returns the regions of a linkSet object.

This method returns the regions corresponding to the bait anchors of a linkSet object.

This method replaces the bait anchors of a linkSet object with new values.

This method replaces the regions of a linkSet object with new values.

This method replaces the anchor1 of a linkSet object with new values.

This method replaces the anchor2 of a linkSet object with new values.

This method replaces the regions of a linkSet object with new values.

This method returns the metadata column of a linkSet object.

This method replaces the metadata column of a linkSet object with new values.

This method returns the names of a linkSet object.

This method replaces the names of a linkSet object.

Usage

```
bait(x) <- value

regions(x) <- value

anchor1(x) <- value

anchor2(x) <- value

unchecked_regions(x) <- value

## S4 method for signature 'linkSet'
anchor1(x)

## S4 method for signature 'linkSet'
anchor2(x)

## S4 method for signature 'linkSet'
regions(x)

## S4 method for signature 'linkSet'
seqinfo(x)

## S4 method for signature 'linkSet'
anchorIds(x, type = "both")

## S4 method for signature 'linkSet'
anchors(x, type = "both", id = FALSE)

## S4 method for signature 'linkSet'
first(x)

## S4 method for signature 'linkSet'
second(x)

## S4 method for signature 'linkSet'
bait(x)

## S4 method for signature 'linkSet'
oe(x)

## S4 method for signature 'linkSet'
regions(x)
```

```
## S4 method for signature 'linkSet'
regionsBait(x)

## S4 replacement method for signature 'linkSet'
bait(x) <- value

## S4 replacement method for signature 'linkSet'
unchecked_regions(x) <- value

## S4 replacement method for signature 'linkSet'
unchecked_anchor1(x) <- value

## S4 replacement method for signature 'linkSet'
unchecked_anchor2(x) <- value

## S4 replacement method for signature 'linkSet'
regions(x) <- value

## S4 replacement method for signature 'linkSet'
regionsBait(x) <- value

## S4 replacement method for signature 'linkSet'
oe(x) <- value

## S4 method for signature 'linkSet'
x$name

## S4 replacement method for signature 'linkSet'
x$name <- value

## S4 method for signature 'linkSet'
names(x)

## S4 replacement method for signature 'linkSet'
names(x) <- value
```

Arguments

x	A linkSet object
value	A character vector of new names
type	The type of anchor to return. Can be "both", "bait", or "oe".
id	If TRUE, returns the anchor IDs instead of the anchors.
name	A character string specifying the name of the metadata column to replace.

Value

For the getters, values in various slots of x are returned, while for the setters, the slots of x are modified accordingly – see Details.

A vector of the regions

A list of anchor IDs.

A list of anchors or anchor IDs.

A GRanges object containing the bait anchors.

A GRanges object containing the oe anchors.

A GRanges object containing the bait anchors.

A GRanges object containing the oe anchors.

A GRanges object containing the regions.

A GRanges object containing the regions corresponding to the bait anchors.

The modified linkSet object with the new bait anchors.

The modified linkSet object with the new regions.

The modified linkSet object with the new anchor1 values.

The modified linkSet object with the new anchor2 values.

The modified linkSet object with the new regions.

The value of the specified metadata column.

The modified linkSet object with the new metadata column value.

A character vector of names

The modified linkSet object with updated names

Author(s)

Gilbert Han

Examples

```
data(linkExample)
anchor1(linkExample)
data(linkExample)
anchor2(linkExample)
data(linkExample)
regions(linkExample)
data(linkExample)
anchorIds(linkExample, type="both")
data(linkExample)
anchors(linkExample, type="both", id=FALSE)
```

---

baitGInteractions *Convert GInteractions to linkSet with bait annotations*

Description

Convert GInteractions with bait range and oe ranges to linkSet

Usage

```
baitGInteractions(x, geneGr, peakGr, ...)

## S4 method for signature 'GInteractions,GRanges,GRanges'
baitGInteractions(x, geneGr, peakGr, geneSymbol = NULL)
```

Arguments

x	A GInteractions object
geneGr	A GRanges object representing genes
peakGr	A GRanges object representing peaks
...	Additional arguments
geneSymbol	A character vector with same length as geneGr or column name in mcols(geneGr) for gene symbols

Value

A linkSet object

Examples

```
# Example usage:
library(GenomicRanges)
library(InteractionSet)

# Create example GRanges objects for genes and peaks
geneGr <- GRanges(seqnames = "chr1",
                  ranges = IRanges(start = c(100, 200), end = c(150, 250)),
                  geneSymbol = c("Gene1", "Gene2"))
peakGr <- GRanges(seqnames = "chr1",
                  ranges = IRanges(start = c(300, 400), end = c(350, 450)))

# Create example GInteractions object
gi <- GInteractions(anchor1 = geneGr, anchor2 = peakGr)

# Convert to linkSet
linkSetObj <- baitGInteractions(gi, geneGr, peakGr, geneSymbol = "geneSymbol")

# Print the linkSet object
print(linkSetObj)
```

---

character_Or_missing-class

Character or Missing Class Union

Description

A class union of character vectors and missing values used in linkSet package for optional character arguments.

Details

This class is used internally by the linkSet package to handle optional character arguments, particularly in the linkSet constructor and methods.

checkPackages *Check if packages are installed for truncated distributions*

Description

Check if required packages are available for truncated distributions

Usage

```
checkPackages(distribution)
```

Arguments

distribution	Character string specifying the distribution type

Value

Logical indicating if packages are available, or stops with error

checkSplitDataNumericalFit *check.split.data.numerical.fit*

Description

Helper function to check if the chicane model can be fit on each element of a split data list.

Check if split data has valid numerical fit for model fitting

Usage

```
checkSplitDataNumericalFit(split.data)
```

Arguments

split.data	Split data for model fitting

Value

Logical indicating if the model can be fit

Check split data numerical fit

None, stops with error if data is invalid

cleanUnusedRegions *Clean Unused Regions*

Description

This function removes unused regions from a linkSet object to minimize memory usage.

Usage

```
cleanUnusedRegions(x)

clean_unused_regions(x)

## S4 method for signature 'linkSet'
cleanUnusedRegions(x)

## S4 method for signature 'linkSet'
clean_unused_regions(x)
```

Arguments

x	A linkSet object

Value

A linkSet object with unused regions removed

Examples

```
data(linkExample)
linkExample <- cleanUnusedRegions(linkExample)
```

Convert,GInteractions-method *Convert GInteractions to linkSet*

Description

Convert other data formats to linkSet. Currently supported: GInteractions, data.frame.

Usage

```
## S4 method for signature 'GInteractions'
Convert(x, baitCol = NULL, ...)

## S4 method for signature 'data.frame'
Convert(x, source = "data.frame", baitCol = "gene", oeCol = "peak", ...)

## S4 method for signature 'Pairs'
Convert(x, baitCol = NULL, ...)
```

```
## S4 method for signature 'ANY'
Convert(x, baitCol = NULL, ...)

readvalidPairs(file, njobs = 1, format = "validPairs")
```

Arguments

x	An object of unsupported class
baitCol	A character string specifying the column to use for bait naming
...	Additional arguments (not used)
source	The source of the data frame, either "data.frame" or "chicane"
oeCol	The column name in the data frame that contains the other end information
file	A character string specifying the path to the validPairs file or 4DN pairs file
njobs	An integer specifying the number of threads to use for reading the file
format	A character string specifying the format of the file, either "validPairs" or "pair". Pair format should be "readID chr1 pos1 chr2 pos2 strand1 strand2". And validPairs should be "readID chr1 pos1 strand1 chr2 pos2 strand2".

Value

A linkSet object

A linkSet object

A linkSet object

Nothing, throws an error

A GInteractions object

Examples

```
library(InteractionSet)
gi <- GInteractions(anchor1 = c(1, 2), anchor2 = c(3, 4),
                    regions = GRanges(seqnames = c("chr1", "chr1", "chr2", "chr2"),
                              ranges = IRanges(start = c(100, 200, 300, 400), width = 50)))
linkset_obj <- Convert(gi)
linkset_obj

df <- data.frame(
  gene = c("gene1", "gene2"),
  peak = c("chr1:1000-2000", "chr2:1500-2500"),
  score = c(5.5, 6.0)
)
linkset_obj <- Convert(df, source = "data.frame", baitCol = "gene", oeCol = "peak")
linkset_obj
```

convertToGrange *Convert string intervals to GRanges*

Description

Convert various interval formats to GRanges objects

Usage

```
convertToGrange(intervals)
```

Arguments

intervals	Interval data to convert

Value

GRanges object

countInteractibility *Count bait and oe interactibility*

Description

This function calculates the number of trans interactions for each bait and oe. The word "interactibility" can refer to https://doi.org/10.1038%2Fnature11279.

Usage

```
countInteractibility(x, baitRegions = TRUE)

## S4 method for signature 'linkSet'
countInteractibility(x, baitRegions = TRUE)
```

Arguments

x	A linkSet object
baitRegions	Whether to count bait regions (default: TRUE)

Details

Count Interaction Interactibility

Value

A linkSet object with counts for each unique interaction

Examples

```
data(linkExample)
linkSet = c(linkExample,linkExample)
linkSet = countInteractions(linkSet)
linkSet = countInteractibility(linkSet)
```

---

countInteractions *Count Bait and Other End Interactions*

Description

This function takes a linkSet object and counts the number of interactions for each bait and other end.

Usage

```
countInteractions(x, baitRegions = TRUE)

## S4 method for signature 'linkSet'
countInteractions(x, baitRegions = TRUE)
```

Arguments

x	A linkSet object
baitRegions	Whether to count bait regions (default: TRUE)

Value

A linkSet object with counts for each unique interaction

Examples

```
data(linkExample)
linkSet = c(linkExample,linkExample)
linkSet = countInteractions(linkSet)
linkSet
```

---

createSampleLinkSet *Create sample linkSet object*

Description

Create a sample linkSet object for testing purposes

Usage

```
createSampleLinkSet()
```

Value

A linkSet object with sample data

`crossGeneEnhancer,linkSet-method`
Cross gene enhancer

Description

Cross gene enhancer

Usage

```
## S4 method for signature 'linkSet'
crossGeneEnhancer(x, score_threshold = NULL)
```

Arguments

`x`	A linkSet object
`score_threshold`	The minimum score to filter interactions

Value

A linkSet object with filtered interactions

Examples

```
data(linkExample)
linkSet = c(linkExample,linkExample)
linkSet = countInteractions(linkSet)
linkSet = filterLinks(linkSet, filter_intra = FALSE, filter_unannotate = FALSE, distance = 100000)
linkSet = crossGeneEnhancer(linkSet, score_threshold = 10)
```

`diagnoseLinkSet,linkSet-method`
Diagnose the linkSet object, return barplot of inter/intra interaction and distance distribution

Description

Diagnose the linkSet object, return barplot of inter/intra interaction and distance distribution

Usage

```
## S4 method for signature 'linkSet'
diagnoseLinkSet(x)
```

Arguments

`x`	A linkSet object

Value

Returns the input linkSet object with additional metadata columns for inter/intra interaction types and distances. Also prints diagnostic plots showing distance distribution and inter/intra interaction proportions.

Examples

```
data(linkExample)
diagnoseLinkSet(linkExample)
```

Embryo_body *Embryo Body BED File*

Description

A compressed BED format file containing genomic regions from mouse embryo body. This dataset contains regulatory elements identified in mouse embryonic development and is provided as example data for demonstrating genomic interaction analysis workflows.

Usage

```
Embryo_body
```

Format

A BED format file with the following columns:

- chromosome: The chromosome name (e.g., chr1, chr2)
- start: The starting position of the feature
- end: The ending position of the feature
- name: Name of the feature
- score: Score between 0 and 1000
- strand: Strand orientation (+ or -)

The file contains 3,727 genomic intervals.

Value

This is a data object. When loaded with `data(Embryo_body)`, it provides access to the file path of the compressed BED file containing embryo body genomic regions.

Source

These data were derived from publicly available mouse embryonic development datasets, specifically from the embryo body, and processed to identify regulatory elements. The original data were processed to create this example dataset for demonstration purposes.

Examples

```
# Get the file path
file_path <- system.file("extdata", "Embryo_body.bed.gz", package = "linkSet")

# Read the file
if (file.exists(file_path)) {
  embryo_data <- read.table(gzfile(file_path),
                            header = FALSE,
                            sep = "\t",
                            stringsAsFactors = FALSE)
  head(embryo_data)
}
```

`enforceOrder` *Enforce order of anchors*

Description

Ensure consistent ordering of anchor pairs

Usage

```
enforceOrder(anchor1, anchor2)
```

Arguments

`anchor1`	First anchor indices
`anchor2`	Second anchor indices

Value

List with ordered anchors

`exportInterBed` *Export linkSet to interBed format*

Description

Exports a linkSet object to a tab-delimited interBed format file

Usage

```
exportInterBed(x, outfile)

## S4 method for signature 'linkSet'
exportInterBed(x, outfile)
```

Arguments

`x`	A linkSet object
`outfile`	Output file path

Details

Export linkSet to interBed Format

Value

None. The function writes to the specified file.

Examples

```
data(linkExample)
tmpfile <- tempfile(fileext = ".txt")
exportInterBed(linkExample, tmpfile)
cat(readLines(tmpfile), sep = "\n")
```

---

exportToLinkSet *Export to linkSet format*

Description

Export to linkSet format

Usage

```
exportToLinkSet(
  cd,
  scoreCol = "score",
  cutoff = 0,
  b2bcutoff = NULL,
  order = c("position", "score")[1],
  removeMT = TRUE
)
```

---

exportWashU *Export linkSet to WashU browser format*

Description

Exports a linkSet object to a tab-delimited format compatible with the WashU genome browser

Usage

```
exportWashU(x, outfile)

## S4 method for signature 'linkSet'
exportWashU(x, outfile)
```

Arguments

x	A linkSet object
outfile	Output file path

Details

Export linkSet to WashU Format

Value

None. The function writes to the specified file.

Examples

```
data(linkExample)
tmpfile <- tempfile(fileext = ".txt")
exportWashU(linkExample, tmpfile)
cat(readLines(tmpfile), sep = "\n")
```

---

filterLinks,linkSet-method

Filter links for further analysis

Description

Filter links for further analysis

Usage

```
## S4 method for signature 'linkSet'
filterLinks(x, filter_intra = TRUE, filter_unannotate = TRUE, distance = NULL)
```

Arguments

x	A linkSet object
filter_intra	Whether to filter intra-chromosomal interactions
filter_unannotate	Whether to filter unannotated interactions
distance	The maximum distance between bait and other end

Value

A linkSet object with filtered interactions

Examples

```
data(linkExample)
linkSet = c(linkExample,linkExample)
linkSet = countInteractions(linkSet)
linkSet = filterLinks(linkSet, filter_intra = FALSE, filter_unannotate = FALSE, distance = 100000)
```

geom_linkset *Add Genome Links to Coverage Plot*

Description

Creates a visualization of genomic links for a linkSet object

Usage

```
geom_linkset(
  linkSet,
  score.col = "count",
  score.threshold = NULL,
  score.color = c("grey70", "#56B1F7", "#132B43"),
  scale.range = 10,
  plot.space = 0.1,
  plot.height = 0.2,
  arrow.size = 0.05,
  remove_x_axis = FALSE,
  link_plot_on_top = FALSE,
  extend.base = 10000,
  show.rect = FALSE,
  x.range = NULL,
  log.scale = TRUE
)

## S4 method for signature 'linkSet'
geom_linkset(
  linkSet,
  score.col = "count",
  score.threshold = NULL,
  score.color = c("grey70", "#56B1F7", "#132B43"),
  scale.range = 10,
  plot.space = 0.1,
  plot.height = 0.2,
  arrow.size = 0.05,
  remove_x_axis = FALSE,
  link_plot_on_top = FALSE,
  extend.base = 1e+06,
  show.rect = FALSE,
  x.range = NULL,
  log.scale = TRUE
)
```

Arguments

linkSet	A linkSet object
score.col	Column name containing score information (default: "count")
score.threshold	Score threshold for filtering links (default: NULL)

score.color	Color vector for score visualization (default: c("grey70", "#56B1F7", "#132B43"))
scale.range	Scale factor for link height (default: 10)
plot.space	Top and bottom margin (default: 0.1)
plot.height	Relative height of link to coverage plot (default: 0.2)
arrow.size	Size of arrow heads (default: 0.05)
remove_x_axis	Whether to remove x-axis (default: FALSE)
link_plot_on_top	Whether to plot links above coverage (default: FALSE)
extend.base	Base pair extension range (default: 10000)
show.rect	Whether to show rectangle borders (default: FALSE)
x.range	Range for x-axis (default: NULL)
log.scale	Whether to use log scale for scores (default: TRUE)

Details

Add Genome Links to Coverage Plot

Value

A ggplot layer object

Examples

```
# Create example linkSet data
gr1 <- GRanges(seqnames = "chr1",
               ranges = IRanges(start = c(1000, 2000), width = 100),
               strand = "+", symbol = c("Gene1", "Gene2"))
gr2 <- GRanges(seqnames = "chr1",
               ranges = IRanges(start = c(5000, 6000), width = 100),
               strand = "+")
linkset_obj <- linkSet(gr1, gr2, specificCol = "symbol")

# Add some metadata for visualization
mcols(linkset_obj)$count <- c(10, 20)

# Example plot (requires ggplot2)

library(ggplot2)
p <- ggplot() + geom_linkset(linkset_obj)
print(p)
```

---

getDistOutput *Get distance output*

Description

Calculate distance metrics for genomic interactions

Usage

```
getDistOutput(regs, ai1, ai2, type, inter_type)
```

Arguments

regs	Genomic regions
ai1	Anchor 1 indices
ai2	Anchor 2 indices
type	Distance type
inter_type	Interaction type

Value

Distance calculations

---

linkExample *Example linkSet Object*

Description

A dataset containing example genomic interactions in linkSet format. This example dataset was created to demonstrate the functionality of the linkSet package for representing and analyzing genomic interactions such as those from Hi-C or promoter-capture Hi-C experiments.

Usage

```
data(linkExample)
```

Format

A linkSet object with example interactions. The object contains:

- Bait regions (anchor1): GRanges object representing promoter regions
- Other end regions (anchor2): GRanges object representing potential enhancer regions
- Metadata columns including: count (interaction strength), baitID (unique identifiers for bait regions), and additional annotations

The data was simulated to reflect typical patterns seen in chromatin interaction data, including distance-dependent interaction frequencies and varying interaction strengths.

Details

The dataset represents simulated chromatin interactions between regulatory elements (enhancers) and promoters across several chromosomes. It includes interaction counts, genomic coordinates for both anchors of the interactions, and associated metadata.

Value

A linkSet object containing example genomic interactions. When loaded with `data(linkExample)`, it provides a linkSet object with simulated chromatin interactions for demonstration and testing purposes.

Source

This is a synthetic dataset created specifically for the linkSet package to demonstrate various analysis workflows. The genomic coordinates are based on the human genome (hg38), but the interaction patterns were simulated.

Examples

```
data(linkExample)
show(linkExample)

# Examine the structure
regions(linkExample)

# View metadata
head(mcols(linkExample))
```

`linkSet` *linkSet: Base Classes for Storing Genomic Link Data*

Description

The linkSet package provides tools for working with genomic link sets, which represent connections between different genomic regions. This package is designed for bioinformatics and genomic data analysis, offering various methods to manipulate and analyze linkSet objects.

Details

The main class provided by this package is the `linkSet` class, which is designed to represent and analyze genomic interactions, particularly focusing on gene-enhancer relationships. Key features include:

- Representation of genomic interactions with two types of anchors: "bait" (typically genes) and "other end" (typically enhancers or other regulatory elements).
- Flexible input methods, supporting construction from various data types.
- Metadata storage for additional information about interactions.
- Integration with Bioconductor classes and tools.
- Methods for annotating promoters and distinguishing between inter- and intra-chromosomal interactions.

Value

This is package documentation. The linkSet package provides classes and methods for working with genomic interaction data. See the individual function documentation for specific return values.

References

Add any relevant references here.

See Also

Useful links:

- https://github.com/GilbertHan1011/linkSet
- Report bugs at https://github.com/GilbertHan1011/linkSet/issues/new

Examples

```
data(linkExample)
linkExample
```

`linkSet,character,GRanges,character_Or_missing-method`
Create a linkSet object from input data

Description

Create a linkSet object from input data

Usage

```
## S4 method for signature 'character,GRanges,character_Or_missing'
linkSet(anchor1, anchor2, specificCol, metadata = list(), ...)
```

Arguments

`anchor1`	For the first method, a character vector of bait names. For the second method, a GRanges object containing anchor1 regions.
`anchor2`	A GRanges object containing anchor2 regions
`specificCol`	Optional character vector specifying names for the baits. Can be either a column name from anchor1's metadata or a vector of names.
`metadata`	Optional list of metadata to store
`...`	Additional columns to add to the linkSet's elementMetadata

Value

A linkSet object containing the interaction data

linkSet,GRanges,GRanges,character_Or_missing-method

Create a linkSet object from input data

Description

Create a linkSet object from input data

Usage

```
## S4 method for signature 'GRanges,GRanges,character_Or_missing'
linkSet(anchor1, anchor2, specificCol, metadata = list(), ...)
```

Arguments

anchor1	For the first method, a character vector of bait names. For the second method, a GRanges object containing anchor1 regions.
anchor2	A GRanges object containing anchor2 regions
specificCol	Optional character vector specifying names for the baits. Can be either a column name from anchor1's metadata or a vector of names.
metadata	Optional list of metadata to store
...	Additional columns to add to the linkSet's elementMetadata

Value

A linkSet object containing the interaction data

linkSet-class *LinkSet object*

Description

The linkSet object is a container for storing gene-enhancer interactions.

Details

The linkSet object is a vectors of paired gene-enhancer interactions.

Slots

nameBait A character vector of the bait names.

anchor1 A integer vector of the first anchor indices.

anchor2 A integer vector of the second anchor indices.

regions A GenomicRanges object of the regions.

NAMES A character vector of the region names.

elementMetadata A DataFrame of the element metadata.

See Also

linkSet

Examples

```
showClass("linkSet")  # shows the known subclasses

set.seed(7000)
N <- 40
all.starts <- round(runif(N, 1, 100))
all.ends <- all.starts + round(runif(N, 5, 20))
all.regions <- GRanges(rep(c("chrA", "chrB"), c(N-10, 10)), IRanges(all.starts, all.ends))
genes = c(rep("SP7",4),rep("ASPN",10),rep("XBP1",6))
Np <- 20
all.anchor1 <- sample(N, Np)
gr1 <- all.regions[all.anchor1]
gr1$symbol <- genes
all.anchor2 <- setdiff(1:40,all.anchor1)
gr2 <- all.regions[all.anchor2]
x <- linkSet(gr1, gr2,specificCol = "symbol")
x
x2 <- linkSet(genes, gr2)
x2
```

---

oe<- *Set Other End (OE) Anchors*

Description

Replace the other end (oe) anchors of a linkSet object with new values

Usage

```
oe(x) <- value
```

Arguments

x	A linkSet object
value	A GRanges object containing the new other end anchors

Details

Set Other End Anchors for linkSet Object

Value

The modified linkSet object

Examples

```
# Create example data
gr1 <- GRanges("chr1", IRanges(1:3, width=1))
gr2 <- GRanges("chr1", IRanges(4:6, width=1))
linkset_obj <- linkSet(gr1, gr2)

# Create new other end anchors
new_oe <- GRanges("chr1", IRanges(7:9, width=1))

# Replace other end anchors
oe(linkset_obj) <- new_oe
```

`orderLinks,linkSet-method`
Order linkSet by mcols

Description

Order linkSet by mcols

Usage

```
## S4 method for signature 'linkSet'
orderLinks(x, by = "count", decreasing = TRUE)
```

Arguments

x	A linkSet object
by	The column name to order by
decreasing	Whether to sort in decreasing order

Value

A linkSet object with ordered interactions

Examples

```
data(linkExample)
linkSet = c(linkExample,linkExample)
linkSet = countInteractions(linkSet)
linkSet = filterLinks(linkSet, filter_intra = FALSE, filter_unannotate = FALSE, distance = 100000)
linkSet = orderLinks(linkSet, by = "count", decreasing = TRUE)
```

pairdist,linkSet-method *Calculate the distance between bait and the other end*

Description

Outputs an integer vector specifying the distance between the interacting bins, depending on the type of distance specified.

Example:

```
rangeA:  |---------|
rangeB:               |---------|
mid:           <----------->
gap:               <-->
span:    <----------------------->
```

- mid: Half the distance between the end of first range and start of second range
- gap: Distance between the end of first range and start of second range
- span: Total span from start of first range to end of second range

Usage

```
## S4 method for signature 'linkSet'
pairdist(x, type = "mid")
```

Arguments

x	A linkSet object
type	The type of distance to calculate, either "mid", "gap", or "span"

Value

A linkSet object with a new metadata column "distance"

Examples

```
data(linkExample)
linkExample <- pairdist(linkExample, type="mid")
```

pasteAnchor *Format anchor information for display*

Description

Format anchor information for display

Usage

```
pasteAnchor(x, append)
```

plotBaits *Plot Baits*

Description

Plot baits in a linkSet object

Usage

```
plotBaits(
  linkset,
  scoreCol = "score",
  countCol = "count",
  n = 4,
  baits = NULL,
  plotBaitNames = TRUE,
  plevel1 = 5,
  plevel2 = 3,
  outfile = NULL,
  width = 20,
  height = 20,
  extend.base = 1e+06,
  bgCol = "black",
  lev2Col = "blue",
  lev1Col = "red",
  ...
)
```

Arguments

linkset	A linkSet object
scoreCol	Column name containing scores for coloring points
countCol	Column name containing counts for y-axis values
n	Number of random baits to plot if baits parameter is NULL
baits	Vector of specific baits to plot. If NULL, n random baits are selected
plotBaitNames	Logical indicating whether to show bait names in plot titles
plevel1	Upper threshold for score coloring (red)
plevel2	Lower threshold for score coloring (blue)
outfile	Output file path. If NULL, plot is displayed rather than saved
width	Width of output plot in inches
height	Height of output plot in inches
extend.base	Base pairs to extend view range on either side of bait
bgCol	Color for points below plevel2 threshold
lev2Col	Color for points between plevel2 and plevel1 thresholds
lev1Col	Color for points above plevel1 threshold
...	Additional plotting parameters

Value

A ggplot object

Examples

```
# Create example linkSet object
library(GenomicRanges)
gr1 <- GRanges(seqnames = c("chr1", "chr1", "chr2"),
               ranges = IRanges(start = c(1000, 2000, 3000), width = 100),
               strand = "+", symbol = c("Gene1", "Gene2", "Gene3"))
gr2 <- GRanges(seqnames = c("chr1", "chr2", "chr2"),
               ranges = IRanges(start = c(5000, 6000, 7000), width = 100),
               strand = "+")
linkset_obj <- linkSet(gr1, gr2, specificCol = "symbol")

# Add score and count metadata for plotting
mcols(linkset_obj)$score <- c(2.5, 4.2, 6.1)
mcols(linkset_obj)$count <- c(10, 25, 15)

# Plot baits (requires annotated bait regions)

# Note: This requires regionsBait to be annotated
# plotBaits(linkset_obj, n = 2)
```

---

plotGenomicRanges *Plot Genomic Ranges*

Description

Creates a visualization of genomic ranges and interactions from a linkSet object

Usage

```
plotGenomicRanges(
  linkset,
  showBait = NULL,
  showOE = NULL,
  x.range = NULL,
  score.col = "count",
  show.rect = TRUE,
  extend.base = 10000,
  ...,
  bait_col = "red",
  oe_col = "DeepSkyBlue3",
  default_col = "grey",
  vjust = NULL,
  linejoin = "mitre",
  na.rm = FALSE,
  minimal_width = 0.01,
  show.legend = NA,
  inherit.aes = TRUE,
```

```
  link_plot_on_top = FALSE,
  arrow.size = 0.05,
  remove_x_axis = FALSE,
  plot.height = 0.4,
  plot.space = 0.1,
  log.scale = TRUE
)

plot_genomic_ranges(
  linkset,
  showBait = NULL,
  showOE = NULL,
  x.range = NULL,
  score.col = "count",
  show.rect = TRUE,
  extend.base = 10000,
  ...,
  bait_col = "red",
  oe_col = "DeepSkyBlue3",
  default_col = "grey",
  vjust = NULL,
  linejoin = "mitre",
  na.rm = FALSE,
  minimal_width = 0.01,
  show.legend = NA,
  inherit.aes = TRUE,
  link_plot_on_top = FALSE,
  arrow.size = 0.05,
  remove_x_axis = FALSE,
  plot.height = 0.4,
  plot.space = 0.1,
  log.scale = TRUE
)

## S4 method for signature 'linkSet'
plot_genomic_ranges(
  linkset,
  showBait = NULL,
  showOE = NULL,
  x.range = NULL,
  score.col = "count",
  show.rect = TRUE,
  extend.base = 10000,
  ...,
  bait_col = "red",
  oe_col = "DeepSkyBlue3",
  default_col = "grey",
  vjust = NULL,
  linejoin = "mitre",
  na.rm = FALSE,
  minimal_width = 0.01,
  show.legend = NA,
```

```
  inherit.aes = TRUE,
  link_plot_on_top = FALSE,
  arrow.size = 0.05,
  remove_x_axis = FALSE,
  plot.height = 0.4,
  plot.space = 0.1,
  log.scale = TRUE
)
```

Arguments

linkset	A linkSet object
showBait	Logical indicating whether to show bait regions (default: NULL)
showOE	Logical indicating whether to show other end regions (default: NULL)
x.range	Range for x-axis (default: NULL)
score.col	Column name containing score information (default: "count")
show.rect	Whether to show rectangle borders (default: TRUE)
extend.base	Base pair extension range (default: 10000)
...	Additional plotting parameters
bait_col	Color for bait regions (default: "red")
oe_col	Color for other end regions (default: "DeepSkyBlue3")
default_col	Default color (default: "grey")
vjust	Vertical adjustment (default: NULL)
linejoin	Line join style (default: "mitre")
na.rm	Whether to remove NA values (default: FALSE)
minimal_width	Minimal width for plotting (default: 0.01)
show.legend	Whether to show legend (default: NA)
inherit.aes	Whether to inherit aesthetics (default: TRUE)
link_plot_on_top	Whether to plot links on top (default: FALSE)
arrow.size	Size of arrow heads (default: 0.05)
remove_x_axis	Whether to remove x-axis (default: FALSE)
plot.height	Relative height of plot (default: 0.4)
plot.space	Plot spacing (default: 0.1)
log.scale	Whether to use log scale (default: TRUE)

Details

Plot Genomic Ranges from linkSet Object

Value

A ggplot object

Examples

```
# Create example linkSet object
library(GenomicRanges)
gr1 <- GRanges(seqnames = c("chr1", "chr1", "chr2"),
               ranges = IRanges(start = c(1000, 2000, 3000), width = 100),
               strand = "+", symbol = c("Gene1", "Gene2", "Gene3"))
gr2 <- GRanges(seqnames = c("chr1", "chr2", "chr2"),
               ranges = IRanges(start = c(5000, 6000, 7000), width = 100),
               strand = "+")
linkset_obj <- linkSet(gr1, gr2, specificCol = "symbol")

# Add count metadata for plotting
mcols(linkset_obj)$count <- c(10, 25, 15)

# Plot genomic ranges (requires annotated bait regions)

# Note: This requires regionsBait to be annotated
# plotGenomicRanges(linkset_obj, extend.base = 1000)
```

---

plotGenomicRanges,linkSet-method

Plot genomic ranges for linkSet objects

Description

This function visualizes the genomic interactions in a linkSet object, showing the bait and other end regions as well as the links between them.

Usage

```
## S4 method for signature 'linkSet'
plotGenomicRanges(
  linkset,
  showBait = NULL,
  showOE = NULL,
  x.range = NULL,
  score.col = "count",
  show.rect = TRUE,
  extend.base = 10000,
  ...,
  bait_col = "red",
  oe_col = "DeepSkyBlue3",
  default_col = "grey",
  vjust = NULL,
  linejoin = "mitre",
  na.rm = FALSE,
  minimal_width = 0.01,
  show.legend = NA,
  inherit.aes = TRUE,
  link_plot_on_top = FALSE,
  arrow.size = 0.05,
```

```
  remove_x_axis = FALSE,
  plot.height = 0.4,
  plot.space = 0.1,
  log.scale = TRUE
)
```

Arguments

linkset	A linkSet object to plot
showBait	Vector of bait regions to display (NULL for all)
showOE	Vector of other end regions to display (NULL for all)
x.range	Range of x-axis to display
score.col	Column name for coloring links
show.rect	Whether to show rectangles for regions
extend.base	Base pairs to extend the plot
...	Additional arguments
bait_col	Color for bait regions
oe_col	Color for other end regions
default_col	Default color
vjust	Vertical justification
linejoin	Line join style
na.rm	Whether to remove NA values
minimal_width	Minimal width for regions
show.legend	Whether to show legend
inherit.aes	Whether to inherit aesthetics
link_plot_on_top	Whether to draw links on top
arrow.size	Size of arrows
remove_x_axis	Whether to remove x axis
plot.height	Height of the plot
plot.space	Space between plots
log.scale	Whether to use log scale for colors

Value

A ggplot object

Examples

```
data(linkExample)
plotGenomicRanges(linkExample, extend.base = 10)
```

reduceRegions *Reduce Regions in a linkSet Object*

Description

This function reduces the bait and/or oe regions of a linkSet object and optionally counts interactions, while maintaining the original length of the linkSet.

Usage

```
reduceRegions(x, region = "both", countInteractions = TRUE, ...)

## S4 method for signature 'linkSet'
reduceRegions(x, region = "both", countInteractions = TRUE, ...)
```

Arguments

x	A linkSet object
region	Character, specifying which regions to reduce: "both", "bait", or "oe" (default: "both")
countInteractions	Logical, whether to count interactions after reducing (default: TRUE)
...	Additional arguments passed to GenomicRanges::reduce

Details

Reduce a linkSet Object

Value

A reduced linkSet object with the same length as the input

Examples

```
data(linkExample)
reduced_linkset <- reduceRegions(linkExample, region = "both", countInteractions = TRUE)
reduced_linkset
```

regionsBait<- *Set Bait Regions*

Description

Replace the regions corresponding to the bait anchors of a linkSet object

Usage

```
regionsBait(x) <- value
```

Arguments

x	A linkSet object
value	A GRanges object containing the new bait regions

Details

Set Bait Regions for linkSet Object

Value

The modified linkSet object

Examples

```
# Create example data
gr1 <- GRanges("chr1", IRanges(1:3, width=1))
gr2 <- GRanges("chr1", IRanges(4:6, width=1))
linkset_obj <- linkSet(gr1, gr2)

# Create new bait regions
new_bait <- GRanges("chr1", IRanges(7:9, width=1))

# Replace bait regions
regionsBait(linkset_obj) <- new_bait
```

---

run_chicane *Run ChICANE Analysis*

Description

This function adapts the chicane function from the ChICANE package to work with the linkSet object format. It runs the full method for detecting significant interactions in capture Hi-C experiments.

Usage

```
run_chicane(linkSet, ...)

## S4 method for signature 'linkSet'
run_chicane(
  linkSet,
  replicate.merging.method = "sum",
  distribution = "negative-binomial",
  include.zeros = "none",
  bait.filters = c(0, 1),
  target.filters = c(0, 1),
  distance.bins = NULL,
  multiple.testing.correction = c("bait-level", "global"),
  adjustment.terms = NULL,
  remove.adjacent = FALSE,
  temp.directory = NULL,
  keep.files = FALSE,
```

```
  maxit = 100,
  epsilon = 1e-08,
  cores = 1,
  trace = FALSE,
  verbose = FALSE
)
```

Arguments

linkSet	A linkSet object containing interaction data
...	Additional arguments passed to methods
replicate.merging.method	Method for merging replicates (default: 'sum')
distribution	Distribution to use for modeling (default: 'negative-binomial')
include.zeros	How to handle zero counts (default: 'none')
bait.filters	Vector of length 2 for bait filtering thresholds (default: c(0,1))
target.filters	Vector of length 2 for target filtering thresholds (default: c(0,1))
distance.bins	Number of distance bins (default: NULL)
multiple.testing.correction	Method for multiple testing correction (default: 'bait-level')
adjustment.terms	Additional terms for model adjustment (default: NULL)
remove.adjacent	Whether to remove adjacent fragments (default: FALSE)
temp.directory	Directory for temporary files (default: NULL)
keep.files	Whether to keep temporary files (default: FALSE)
maxit	Maximum iterations for model fitting (default: 100)
epsilon	Convergence threshold (default: 1e-8)
cores	Number of CPU cores to use (default: 1)
trace	Whether to print trace information (default: FALSE)
verbose	Whether to print progress information (default: FALSE)

Details

Run ChICANE Analysis on linkSet Object

Value

A linkSet object with additional columns:

- expected The expected number of reads linking fragments under the fitted model
- p.value P-value for test of observed vs expected read counts
- q.value FDR-corrected p-value

Examples

```
# Create example data
gr1 <- GRanges(seqnames = c("chr1", "chr3", "chr3"),
               ranges = IRanges(start = c(1000, 2000, 3000), width = 100),
               strand = "+", symbol = c("BRCA1", "TP53", "NONEXISTENT"))
gr2 <- GRanges(seqnames = c("chr1", "chr2", "chr3"),
               ranges = IRanges(start = c(5000, 6000, 7000), width = 100),
               strand = "+")
ls <- linkSet(gr1, gr2, specificCol = "symbol")

# Annotate and prepare data
annotated_ls <- suppressWarnings(
  annotatePromoter(ls, genome = "hg38", upstream = 500, overwrite = TRUE)
)
annotated_ls <- countInteractibility(annotated_ls)
annotated_ls <- linkSet::pairdist(annotated_ls)

# Run analysis
result_ls <- run_chicane(
  annotated_ls,
  replicate.merging.method = 'sum',
  bait.filters = c(0, 1),
  target.filters = c(0, 1),
  distance.bins = NULL,
  multiple.testing.correction = 'bait-level',
  verbose = TRUE
)
```

---

show,linkSet-method *Display detailed information about a linkSet object*

Description

Display detailed information about a linkSet object

Usage

```
## S4 method for signature 'linkSet'
show(object)
```

Arguments

object	A linkSet object to display

Value

Invisibly returns NULL. This method is called for its side effect of printing detailed information about the linkSet object to the console.

Examples

```
# Example usage of show method for linkSet object
gr1 <- GRanges(seqnames = c("chr1", "chr2", "chr3"),
               ranges = IRanges(start = c(1000, 2000, 3000), width = 100),
               strand = "+", symbol = c("BRCA1", "TP53", "NONEXISTENT"))
gr2 <- GRanges(seqnames = c("chr1", "chr2", "chr3"),
               ranges = IRanges(start = c(5000, 6000, 7000), width = 100),
               strand = "+")
ls <- linkSet(gr1, gr2, specificCol = "symbol")
show(ls)
```

---

showLinkSet *Show linkSet Object Details*

Description

Displays detailed information about a linkSet object, including regions, metadata, and optionally sequence information.

Usage

```
showLinkSet(
  object,
  margin = "",
  print.seqinfo = FALSE,
  print.classinfo = FALSE,
  baitRegion = FALSE,
  ...
)

## S4 method for signature 'linkSet'
showLinkSet(
  object,
  margin = "",
  print.seqinfo = FALSE,
  print.classinfo = FALSE,
  baitRegion = FALSE
)
```

Arguments

`object`	A linkSet object to display
`margin`	Character string for display margin (default: "")
`print.seqinfo`	Logical, whether to print sequence information (default: FALSE)
`print.classinfo`	Logical, whether to print class information (default: FALSE)
`baitRegion`	Logical, whether to display bait regions (default: FALSE)
`...`	Additional arguments

Details

Display Detailed Information About a linkSet Object

Value

None (invisible NULL)

Examples

```
gr1 <- GRanges(seqnames = c("chr1", "chr2", "chr3"),
               ranges = IRanges(start = c(1000, 2000, 3000), width = 100),
               strand = "+", symbol = c("BRCA1", "TP53", "NONEXISTENT"))
gr2 <- GRanges(seqnames = c("chr1", "chr2", "chr3"),
               ranges = IRanges(start = c(5000, 6000, 7000), width = 100),
               strand = "+")
linkset_obj <- linkSet(gr1, gr2, specificCol = "symbol")
showLinkSet(linkset_obj)
```

---

subsetBait,linkSet-method

Subset linkSet object based on bait names

Description

Subset linkSet object based on bait names

Subset linkSet object based on bait regions

Subset linkSet object based on other end (oe) regions

Usage

```
## S4 method for signature 'linkSet'
subsetBait(x, subset)

## S4 method for signature 'linkSet'
subsetBaitRegion(x, subset)

## S4 method for signature 'linkSet'
subsetOE(x, subset)
```

Arguments

x	A linkSet object
subset	A GRanges object specifying the regions to keep

Value

A new linkSet object containing only the specified bait interactions

A new linkSet object containing only the interactions with bait regions overlapping the subset

A new linkSet object containing only the interactions with oe regions overlapping the subset

Examples

```
data(linkExample)
subset_bait_names <- c("bait1", "bait2")
subsetted_linkSet <- subsetBait(linkExample, subset_bait_names)
data(linkExample)
subset_bait_regions <- GRanges(seqnames = "chr1",
                               ranges = IRanges(start = c(100, 200), end = c(150, 250)))
subsetted_linkSet <- subsetBaitRegion(linkExample, subset_bait_regions)
data(linkExample)
subset_oe_regions <- GRanges(seqnames = "chr1",
                             ranges = IRanges(start = c(300, 400), end = c(350, 450)))
subsetted_linkSet <- subsetOE(linkExample, subset_oe_regions)
```

---

themeLinkset *Theme for linkSet plots*

Description

Theme for linkSet plots

Usage

```
themeLinkset(x.range, margin.len, show.rect)
```

Arguments

x.range	The x-axis range
margin.len	Margin length
show.rect	Whether to show rectangle

Value

A ggplot2 theme

---

themeRange *Theme for genomic range plots*

Description

Theme for genomic range plots

Usage

```
themeRange(x.range, show.rect)
```

Arguments

x.range	The x-axis range
show.rect	Whether to show rectangle

Value

A ggplot2 theme

`trim,linkSet-method` *linkSet-GRange-Methods*

Description

This man page documents intra range transformations of a linkSet object.

Usage

```
## S4 method for signature 'linkSet'
trim(x, use.names = TRUE, ...)

## S4 method for signature 'linkSet'
resize(x, width, fix = "start", use.names = TRUE, ...)

## S4 method for signature 'linkSet'
resizeRegions(
  x,
  width = 1000,
  fix = "start",
  use.names = TRUE,
  region = "both",
  ...
)

## S4 method for signature 'linkSet'
narrow(x, start = NA, end = NA, width = NA, use.names = TRUE)

## S4 method for signature 'linkSet'
narrowRegions(
  x,
  start = NA,
  end = NA,
  width = NA,
  use.names = TRUE,
  region = "both"
)

## S4 method for signature 'linkSet'
shift(x, shift = 0L, use.names = TRUE)

## S4 method for signature 'linkSet'
shiftRegions(x, shift = 0L, use.names = TRUE, region = "both")

## S4 method for signature 'linkSet'
flank(
  x,
```

```
  width,
  start = TRUE,
  both = FALSE,
  use.names = TRUE,
  ignore.strand = FALSE
)

## S4 method for signature 'linkSet'
flankRegions(
  x,
  width,
  start = TRUE,
  both = FALSE,
  use.names = TRUE,
  ignore.strand = FALSE,
  region = "both"
)

## S4 method for signature 'linkSet'
promoters(x, upstream = 2000, downstream = 200, use.names = TRUE)

## S4 method for signature 'linkSet'
promoterRegions(
  x,
  upstream = 2000,
  downstream = 200,
  use.names = TRUE,
  region = "both"
)

## S4 method for signature 'linkSet'
width(x)

## S4 method for signature 'linkSet'
reduce(x, drop.empty.ranges = FALSE, ...)
```

Arguments

x	A linkSet object
use.names	A logical indicating whether to use names
...	Additional arguments passed to the GenomicRanges trim method
width	The desired width of the output ranges
fix	The anchor point for resizing operations ("start", "end", or "center")
region	Which regions to modify ("both", "bait", or "oe")
start, end	The desired start and end coordinates for narrowing
shift	The number of positions to shift
both	Whether to get flanking regions on both sides
ignore.strand	TRUE or FALSE. Whether the strand of the input ranges should be ignored or not. See details below.

upstream, downstream
: Number of bases upstream/downstream for promoter regions

drop.empty.ranges
: Whether to drop empty ranges when reducing

Value

A linkSet object

Author(s)

Gilbert Han

Examples

```
data(linkExample)
resize_bait <- resizeRegions(linkExample, width = 75, fix = "start", region = "bait")
resize_bait

narrow_bait <- narrowRegions(linkExample, start = 1, width = 5, region = "bait")
narrow_bait

shift_oe <- shiftRegions(linkExample, shift = 10, region = "oe")
shift_oe

flank_bait <- flankRegions(linkExample, width = 100, start = TRUE, both = FALSE,
                           use.names = TRUE, ignore.strand = FALSE, region = "bait")
flank_bait

width(linkExample)
```

---

unchecked_anchor1<- *Set unchecked anchor1*

Description

Set unchecked anchor1

Usage

```
unchecked_anchor1(x) <- value
```

---

unchecked_anchor2<- *Set unchecked anchor2*

Description

Set unchecked anchor2

Usage

```
unchecked_anchor2(x) <- value
```

verifyLinkSet *verify.linkSet*

Description

Verify that linkSet object is in expected format. Throws an error if object does not fit requirements.

Check if an error matches the error raised by \code{glm.nb} due to an inflated theta estimate.

This happens when the variance of the negative binomial does not exceed the mean (i.e. there is no overdispersion). In such cases, the Poisson distribution may be a suitable alternative.

Check if chicane model can be fit on a given dataset.

glm.nb does not work when all responses are constant, or there are only two unique values and a covariate is a perfect predictor.

Perform multiple testing correction on p-values from interaction test. By default, multiple testing correction is applied per bait. To change this to a global multiple testing correction, set bait.level = FALSE.

Split a data frame into a prespecified number of bins, using split and cut. Unlike the default R functions, this does not fail when asked to split the data into a single bin.

Verify that interaction.data object is in expected format. Throws an error if object does not fit requirements.

Fit GLM according to a specified distribution. This needs to be done separately from glm in order to include negative binomial and truncated distributions as options.

Check that the model fit contains the same number of rows as the data used to fit it,

and throw an error if not

Check if a warning object is an iteration limit reached warning from glm.nb

Check if a warning matches the square root warning raised by \code{glm.nb} due to an inflated theta es

This happens when the variance of the negative binomial does not exceed the mean (i.e. there is no overdispersion). In such cases, the Poisson distribution may be a suitable alternative.

Usage

```
verifyLinkSet(linkSet)

isGlmNbThetaError(e)

checkModelNumericalFit(interaction.data)

multipleTestingCorrect(interaction.data, bait.level = TRUE)

smartSplit(dat, bins)
```

```
verifyInteractionData(interaction.data)

fitGlm(
  formula,
  data,
  distribution = c("negative-binomial", "poisson", "truncated-poisson",
    "truncated-negative-binomial"),
  start = NULL,
  init.theta = NULL,
  maxit = 100,
  epsilon = 1e-08,
  trace = FALSE
)

modelRowsSanityCheck(model.data, model)

isGlmNbMaxiterWarning(w)

isGlmNbThetaWarning(w)
```

Arguments

linkSet	Object to be verified.
e	Error object
interaction.data	Object to be verified.
bait.level	Logical indicating whether multiple testing correction should be performed per bait.
dat	Data frame or data table to be split
bins	Number of bins to split data into
formula	Formula specifying model of interest
data	Data frame containing variables specified in formula
distribution	Name of distribution of the counts. Options are 'negative-binomial', 'poisson', 'truncated-poisson', and 'truncated-negative-binomial'
start	Starting values for model coefficients
init.theta	Initial value of theta if fitting the negative binomial distribution
maxit	Maximum number of IWLS iterations for fitting the model (passed to glm.control)
epsilon	Positive convergence tolerance for Poisson and negative binomial models. Passed to glm.control
trace	Logical indicating if output should be produced for each of model fitting procedure. Passed to glm.control or gamlss.control
model.data	Data used to fit model
model	Resulting negative binomial model object
w	Warning object

Value

None

Boolean indicating if error matches

boolean indicating if model can be fit

Original data table with new column

`q.value`	FDR-corrected p-value

List with `bins` elements. Each element corresponds to one portion of the data

None

List with elements

`model`	model object
`expected.values`	vector of expected values for each element in original data
`p.values`	vector of p-values for test of significantly higher response than expected

None

Logical indicating if warning matches iteration limit reached warning

Boolean indicating if warning matches

`withTxDb` *Database Operation with Connection Management*

Description

Executes a database operation while managing the connection lifecycle automatically.

Usage

```
withTxDb(x, expr, ...)

## S4 method for signature 'character,function'
withTxDb(x, expr, ...)
```

Arguments

`x`	Character string specifying the genome ("hg38", "hg19", or "mm10")
`expr`	Function to execute with database connection
`...`	Additional arguments passed to expr

Details

Execute Database Operation with Automatic Connection Management

Value

Result of the database operation

Examples

```
# Example 1: Get genes from hg38
result <- withTxDb("hg38", function(src) {
  genes <- Organism.dplyr::genes(src)
  return(head(genes))
})

# Example 2: Get transcripts
result2 <- withTxDb("hg38", function(src) {
  transcripts <- Organism.dplyr::transcripts(src)
  return(head(transcripts))
})
```

Index