Package 'CNEr'

October 12, 2016

Version 1.8.3

Date 2016-04-12

Title CNE Detection and Visualization

Description Large-scale identification and advanced visualization of sets of conserved noncoding elements.

Author Ge Tan <ge.tan09@imperial.ac.uk>

Maintainer Ge Tan <ge.tan09@imperial.ac.uk>

Imports Biostrings(>= 2.33.4), RSQLite(>= 0.11.4), GenomeInfoDb(>= 1.1.3), GenomicRanges(>= 1.23.16), rtracklayer(>= 1.25.5), XVector(>= 0.5.4), DBI(>= 0.2-7), GenomicAlignments(>= 1.1.9), methods, S4Vectors(>= 0.9.25), IRanges(>= 2.5.27), readr(>= 0.2.2), BiocGenerics, tools, parallel

Depends R (>= 3.2.2)

Suggests Gviz(>= 1.7.4), BiocStyle, knitr, rmarkdown, testthat

LinkingTo S4Vectors, IRanges, XVector

VignetteBuilder knitr

License GPL-2 | file LICENSE

License_restricts_use yes

URL https://github.com/ge11232002/CNEr

BugReports https://github.com/ge11232002/CNEr/issues

Type Package

biocViews GeneRegulation, Visualization, DataImport

NeedsCompilation yes

LazyData no

Collate GRangePairs-class.R AllGenerics.R AllClasses.R utils.R ceScan.R plot.R makeGeneDbFromUCSC.R io.R scoringMatrix.R subAxt-methods.R Axt-methods.R DB.R AssemblyStats.R GRB.R WholeGenomeAlignment.R

R topics documented:

axisTrack *Example data for plotting annotation.*

Description

Five annotation tracks for plotting in Gviz.

Usage

```
data(axisTrack)
data(cpgIslands)
data(refGenes)
data(ideoTrack)
```

Details

These tracks are based on genome="hg19", chr = "chr11", start = 31000000L, end = 33000000L.

Examples

```
data(axisTrack)
data(cpgIslands)
data(refGenes)
data(ideoTrack)
```

Axt-class *Class "Axt"*

Description

The Axt S4 object to hold a axt file.

Usage

```
## Constructors:
Axt(targetRanges=GRanges(), targetSeqs=DNAStringSet(),
    queryRanges=GRanges(), querySeqs=DNAStringSet(),
    score=integer(0), symCount=integer(0))

## Accessor-like methods:
## S4 method for signature 'Axt'
targetRanges(x)
## S4 method for signature 'Axt'
targetSeqs(x)
## S4 method for signature 'Axt'
queryRanges(x)
```

```
## S4 method for signature 'Axt'
querySeqs(x)
## S4 method for signature 'Axt'
score(x)
## S4 method for signature 'Axt'
symCount(x)
## S4 method for signature 'Axt'
nchar(x)
## ... and more (see Methods)
```

Arguments

`targetRanges`	Object of class `"GRanges"`: The ranges of net alignments on reference genome.
`targetSeqs`	Object of class `"DNAStringSet"`: The alignment sequences of reference genome.
`queryRanges`	Object of class `"GRanges"`: The ranges of net alignments on query genome.
`querySeqs`	Object of class `"DNAStringSet"`: The alignment sequences of query genome.
`score`	Object of class `"integer"`: The alignment score.
`symCount`	Object of class `"integer"`: The alignment length.
`x`	Object of class `"Axt"`: A Axt object.

Methods

[`signature(x = "Axt", i = "ANY", j = "ANY")`: Axt getter

c `signature(x = "Axt")`: Axt concatenator.

length `signature(x = "Axt")`: Get the number of alignments.

queryRanges `signature(x = "Axt")`: Get the ranges of query genome.

querySeqs `signature(x = "Axt")`: Get the alignment sequences of query genome.

score `signature(x = "Axt")`: Get the alignment score.

symCount,nchar `signature(x = "Axt")`: Get the alignment lengths.

targetRanges `signature(x = "Axt")`: Get the ranges of reference genome.

targetSeqs `signature(x = "Axt")`: Get the alignment sequences of reference genome.

Author(s)

Ge Tan

See Also

`readAxt` `writeAxt` `subAxt`

Examples

```
showClass("Axt")
```

axtChain *axtChain*

Description

Wrapper function of axtChain: chain together psl alignments. If two matching alignments next to each other are close enough, they are joined into one fragment. This function doesn't work on Windows platform since Kent utilities only support Linux and Unix platform.

Usage

```
axtChain(psls, chains=sub("\\.psl$", ".chain", psls, ignore.case=TRUE),
         assemblyTarget, assemblyQuery,
         distance=c("far", "medium", "far"),
         removePsl=TRUE, binary="axtChain")
```

Arguments

psls	character(n): file names of input *psl* files.
chains	character(n): file names of output *chain* files. By default, in the same folder of input lav files with same names.
assemblyTarget	character(1): the file name of target assembly *twoBit* file.
assemblyQuery	character(1): the file name of query assembly *twoBit* file.
distance	It can be "far", "medium" or "close". It decides the score matrix used in *lastz* aligner. See '?scoringMatrix' for more details.
removePsl	boolean: When TRUE, the input *psl* files will be removed from the conversion.
binary	character(1): the name/filename of the binary axtChain to call.

Value

character(n): the file names of output *chain* files.

Author(s)

Ge Tan

References

http://hgdownload.cse.ucsc.edu/admin/exe/

See Also

lavToPsl

Examples

```
## Not run:
## This example doesn't run because it requires two bit files and external
## Kent utilities.
  psls <- tools::list_files_with_exts(
            dir="/Users/gtan/OneDrive/Project/CSC/CNEr/axt", exts="psl")
  assemblyTarget <- "/Users/gtan/OneDrive/Project/CSC/CNEr/2bit/danRer10.2bit"
  assemblyQuery <- "/Users/gtan/OneDrive/Project/CSC/CNEr/2bit/hg38.2bit"
  axtChain(psls, assemblyTarget=assemblyTarget,
           assemblyQuery=assemblyQuery, distance="far",
           removePsl=FALSE, binary="axtChain")

## End(Not run)
```

---

axtDanRer7Hg19 *The dataset axtDanRer7Hg19, axtHg19DanRer7*

Description

The example CNEs from part of hg19 and danRer7 comparison.

Usage

```
data(axtDanRer7Hg19)
       data(axtHg19DanRer7)
```

Examples

```
data(axtDanRer7Hg19)
```

---

axtInfo *axtInfo function*

Description

Given the path of axt file, retrieve the alignments' withs information.

Usage

```
  axtInfo(axtFiles)
```

Arguments

axtFiles	The filenames of axt files.

Value

A vector of `integer` is returned. It stores the withds of all the alignments.

Author(s)

Ge Tan

See Also

`readAxt`

Examples

```
axtFilesHg19DanRer7 <- file.path(system.file("extdata", package="CNEr"),
                                 "hg19.danRer7.net.axt")
axtInfoHg19DanRer7 <- axtInfo(axtFilesHg19DanRer7)
```

---

binning-utils *UCSC bin indexing system utility functions*

Description

Utility functions for UCSC bin indexing system manipulation

Usage

```
binFromCoordRange(starts, ends)
binRangesFromCoordRange(start, end)
binRestrictionString(start, end, field="bin")
```

Arguments

`starts, ends`	A vector of integers. A set of ranges.
`start, end`	A integer vector of length 1. A coordinate range.
`field`	Name of bin column. Default: "bin".

Details

The UCSC bin indexing system was initially suggested by Richard Durbin and Lincoln Stein to speed up the SELECT of a SQL query for the rows overlapping with certain genome coordinate. The system first used in UCSC genome browser is described by Kent et. al. (2002).

Value

For `binFromCoordRange`, it returns the bin number that should be assigned to a feature spanning the given range. Usually it is used when creating a database for the features.

For `binRangesFromCoordRange`, it returns the set of bin ranges that overlap a given coordinate range. It is usually used to find out the bins overlapped with a range. For SQL query, it is more convenient to use `binRestrictionString` than to use this function directly.

For `binRestrictionString`, it returns a string to be used in the WHERE section of a SQL SELECT statement that is to select features overlapping a certain range. * USE THIS WHEN QUERYING A DB *

Author(s)

Ge Tan

References

Kent, W. J., Sugnet, C. W., Furey, T. S., Roskin, K. M., Pringle, T. H., Zahler, A. M., & Haussler, A. D. (2002). The Human Genome Browser at UCSC. Genome Research, 12(6), 996-1006. doi:10.1101/gr.229102

http://genomewiki.ucsc.edu/index.php/Bin_indexing_system

Examples

```
binFromCoordRange(starts=c(10003, 1000000), ends=c(10004, 1100000))
binRangesFromCoordRange(start=10000, end=2000000)
binRestrictionString(start=10000, end=2000000, field="bin")
```

---

blatCNE *Wrapper function of blat for CNEs*

Description

This wrapper function blat the CNEs against the reference genome.

Usage

```
blatCNE(CNE, winSize, cutoffs1, cutoffs2, assembly1Twobit, assembly2Twobit,
        blatOptions=NULL, cutIdentity=90, tmpDir=tempdir(), blatBinary="blat")
```

Arguments

`CNE`	A object of `data.frame`. Usually it is generated from `cneMerge` function.
`winSize`	A object of `integer`. The window size used for identifying the CNEs, such as 50 or 30.
`cutoffs1, cutoffs2`	A object of `integer`. The CNEs with more than the cutoff hits on the reference genome are removed.

`assembly1Twobit, assembly2Twobit`
: A object of `character`. The path of reference genome in two bit file format.

`blatOptions`
: A object of `character`. When it is NULL, a bunch of preset parameters for blat will be given based on the winSize parameter.

`cutIdentity`
: A object of `integer`. Sets minimum sequence identity (in percent) in blat. Default is 90.

`tmpDir`
: A object of `character`. By default, the R's temp dir is used. You can specify other path if your R's temp dir is small.

`blatBinary`
: A object of `character`. The path of blat binary.

Details

When winSize > 45, the blat options is "-tileSize=11 -minScore=30 -repMatch=1024".

When 35 < winSize <= 45, the blat options is "-tileSize=10 -minScore=28 -repMatch=4096".

When the winSize <= 35, the blat options is "-tileSize=9 -minScore=24 -repMatch=16384".

Value

A `data.frame` containing the CNEs is returned.

Author(s)

Ge Tan

Examples

```
## Not run:
  assemblyHg19Twobit = "/Users/gtan/CSC/CNEr/2bit/hg19.2bit"
  assemblyDanRer7Twobit = "/Users/gtan/CSC/CNEr/2bit/danRer7.2bit"
  cneBlatedDanRer7Hg19 = list()
  for(i in 1:length(cneMergedDanRer7Hg19)){
    cneBlatedDanRer7Hg19[[names(cneMergedDanRer7Hg19)[i]]] =
    blatCNE(cneMergedDanRer7Hg19[[i]],
      as.integer(sub("\d+_", "", names(cneMergedDanRer7Hg19)[i])),
      cutoffs1=4L, cutoffs2=8L,
      assembly1Twobit=assemblyDanRer7Twobit,
      assembly2Twobit=assemblyHg19Twobit,
      blatBinary="blat")
    }

## End(Not run)
```

ceScan-methods *ceScan function*

Description

This is the main function for conserved noncoding elements (CNEs) identification.

Usage

```
ceScan(axts, tFilter, qFilter, qSizes, thresholds="49_50")
```

Arguments

axts	A Axt object or character object with the paths of axt files.
tFilter	A GRanges object or character object with the path of bed file for target genome filter. This argument can also be missing when target filter is not available.
qFilter	A GRanges object or character object with the path of bed file for query genome filter. This argument can also be missing when query filter is not available.
qSizes	A Seqinfo object which contains the seqnames and seqlengths for query genome. This argument can be missing when qFilter is missing.
thresholds	A character object specifiying the scanning windows and minimal score. It can be specified in th form of "45_50" with scanning windows 50 and minial score 45. More than one thresholds can be provided.

Details

ceScan scan the axts alignmnets and identify the CNEs. ceScan can accept axts in Axt object and filter in GRanges object, or directly the axt files and bed files. When the axt files and bed files are ready for computation, it is recommended to use them directly rather than read them into R first.

The details of algorithm will given in the vignette.

Value

A list of data.frame is returned. Each element of the list is for one threshold.

Methods

```
signature(axts = "Axt", tFilter = "GRanges", qFilter = "GRanges", qSizes = "Seqinfo")

signature(axts = "Axt", tFilter = "GRanges", qFilter = "missing", qSizes = "missing")

signature(axts = "Axt", tFilter = "missing", qFilter = "GRanges", qSizes = "Seqinfo")
```

```
signature(axts = "Axt", tFilter = "missing", qFilter = "missing", qSizes = "missing")

signature(axts = "character", tFilter = "character", qFilter = "character", qSizes = "Seqinfo")

signature(axts = "character", tFilter = "character", qFilter = "missing", qSizes = "missing")

signature(axts = "character", tFilter = "missing", qFilter = "character", qSizes = "Seqinfo")

signature(axts = "character", tFilter = "missing", qFilter = "missing", qSizes = "missing")
```

Author(s)

Ge Tan

Examples

```
axtFilesHg19DanRer7 = file.path(system.file("extdata", package="CNEr"),
                                "hg19.danRer7.net.axt")
axtHg19DanRer7 = readAxt(axtFilesHg19DanRer7)
axtFilesDanRer7Hg19 = file.path(system.file("extdata", package="CNEr"),
                                "danRer7.hg19.net.axt")
axtDanRer7Hg19 = readAxt(axtFilesDanRer7Hg19)
bedHg19Fn = file.path(system.file("extdata", package="CNEr"),
                      "filter_regions.hg19.bed")
bedHg19 = readBed(bedHg19Fn)
bedDanRer7Fn = file.path(system.file("extdata", package="CNEr"),
                         "filter_regions.danRer7.bed")
bedDanRer7 = readBed(bedDanRer7Fn)
qSizesHg19 = fetchChromSizes("hg19")
qSizesDanRer7 = fetchChromSizes("danRer7")
CNEHg19DanRer7 = ceScan(axts=axtHg19DanRer7, tFilter=bedHg19,
                        qFilter=bedDanRer7, qSizes=qSizesDanRer7,
                        thresholds=c("45_50", "48_50", "49_50"))
CNEDanRer7Hg19 = ceScan(axts=axtDanRer7Hg19, tFilter=bedDanRer7,
                        qFilter=bedHg19, qSizes=qSizesHg19,
                        thresholds=c("45_50", "48_50", "49_50"))
```

---

ceScanOneStep *ceScanOneStep function*

Description

This function run cne detection in one function.

Usage

```
ceScanOneStep(axt1, filter1=NULL, sizes1, assembly1, twoBit1,
              axt2, filter2=NULL, sizes2, assembly2, twoBit2,
              thresholds=c("49_50"), blatBinary="blat",
              blatCutoff1, blatCutoff2)
```

Arguments

axt1,axt2	The axt object or axt filenames with each assembly as referecne.
filter1,filter2	The GRanges object or bed filenames.
sizes1,sizes2	A Seqinfo object which contains the seqnames and seqlengths for each assembly.
assembly1,assembly2	The assembly names.
twoBit1,twoBit2	The file names of two bit files of two assemblies.
thresholds	A character object specifiying the scanning windows and minimal score. It can be specified in th form of "45_50" with scanning windows 50 and minial score 45. More than one thresholds can be provided.
blatBinary	A object of character. The path of blat binary.
blatCutoff1, blatCutoff2	A object of integer. The CNEs with more than the cutoff hits on the reference genome are removed.

Value

An object CNE is returned.

Author(s)

Ge Tan

---

chainMergeSort *chainMergeSort*

Description

Wrapper function of chainMergeSort: Combine sorted files into larger sorted file. This function doesn't work on Windows platform since Kent utilities only support Linux and Unix platform.

Usage

```
chainMergeSort(chains, assemblyTarget, assemblyQuery,
               allChain=paste0(sub("\\.2bit$", "", basename(assemblyTarget),
                                   ignore.case=TRUE), ".",
                               sub("\\.2bit$", "", basename(assemblyQuery),
                                   ignore.case=TRUE), ".all.chain"),
               removeChains=TRUE, binary="chainMergeSort")
```

Arguments

`chains`	`character`(n): file names of input *chains* files.
`assemblyTarget`	`character`(1): the file name of target assembly *twoBit* file.
`assemblyQuery`	`character`(1): the file name of query assembly *twoBit* file.
`allChain`	`character`(1): file names of merged *allChain* file.
`removeChains`	`boolean`: When `TRUE`, the input *chains* files will be removed after the conversion.
`binary`	`character`(1): the name/filename of the binary `chainMergeSort` to call.

Details

This *allChain* file is what we get from UCSC download, *e.g.*, **hg19.danRer7.all.chain.gz**.

Value

`character`(1): the file names of merged *allChain* file.

Author(s)

Ge Tan

References

http://hgdownload.cse.ucsc.edu/admin/exe/

See Also

axtChain

Examples

```
## Not run:
## This example doesn't run because it requires two bit files and external
## Kent utilities.
  chains <- tools::list_files_with_exts(
              dir="/Users/gtan/OneDrive/Project/CSC/CNEr/axt", exts="chain")
  assemblyTarget <- "/Users/gtan/OneDrive/Project/CSC/CNEr/2bit/danRer10.2bit"
  assemblyQuery <- "/Users/gtan/OneDrive/Project/CSC/CNEr/2bit/hg38.2bit"
  chainMergeSort(chains, assemblyTarget, assemblyQuery,
    allChain=file.path("/Users/gtan/OneDrive/Project/CSC/CNEr/axt",
```

```
                        paste0(sub("\\.2bit$", "", basename(assemblyTarget),
                                   ignore.case=TRUE), ".",
                               sub("\\.2bit$", "", basename(assemblyQuery),
                                   ignore.case=TRUE), ".all.chain")),
      removeChains=FALSE, binary="chainMergeSort")

## End(Not run)
```

---

chainNetSyntenic *chainNetSyntenic*

Description

Wrapper function of *chainNetSyntenic*: Make alignment nets out of chains and add synteny info to net. This function doesn't work on Windows platform since Kent utilities only support Linux and Unix platform.

Usage

```
chainNetSyntenic(allPreChain, assemblyTarget, assemblyQuery,
                 netSyntenicFile=paste0(sub("\\.2bit$", "",
                                            basename(assemblyTarget),
                                            ignore.case = TRUE), ".",
                                        sub("\\.2bit$", "",
                                            basename(assemblyQuery),
                                            ignore.case = TRUE),
                                            ".noClass.net"),
                 binaryChainNet="chainNet", binaryNetSyntenic="netSyntenic")
```

Arguments

`allPreChain`	`character(1)`: file names of input *allPreChain* file.
`assemblyTarget`	`character(1)`: the file name of target assembly *twoBit* file.
`assemblyQuery`	`character(1)`: the file name of query assembly *twoBit* file.
`netSyntenicFile`	`character(1)`: file names of output *netSyntenicFile* file.
`binaryChainNet`	`character(1)`: the name/filename of the binary `chainNet` to call.
`binaryNetSyntenic`	`character(1)`: the name/filename of the binary `netSyntenic` to call.

Details

Add classification information using the database tables: actually this step is not necessary in this pipeline according to http://blog.gmane.org/gmane.science.biology.ucscgenome.general/month=20130301. The class information will only be used for Genome Browser. Since it needs some specific modification of the table names for certain species, we skip this step now. If this step is done, then the generated *class.net* is the gzipped net file that you see in UCSC Downloads area.

Value

character(1): the file names of generated *net* file.

Author(s)

Ge Tan

References

http://hgdownload.cse.ucsc.edu/admin/exe/

See Also

chainPreNet

Examples

```
## Not run:
## This example doesn't run because it requires two bit files and external
## Kent utilities.
  allPreChain <- file.path("/Users/gtan/OneDrive/Project/CSC/CNEr/axt",
                           "danRer10.hg38.all.pre.chain")
  assemblyTarget <- "/Users/gtan/OneDrive/Project/CSC/CNEr/2bit/danRer10.2bit"
  assemblyQuery <- "/Users/gtan/OneDrive/Project/CSC/CNEr/2bit/hg38.2bit"
  chainNetSyntenic(allPreChain, assemblyTarget, assemblyQuery,
                   netSyntenicFile=file.path(
                     "/Users/gtan/OneDrive/Project/CSC/CNEr/axt",
                                      paste0(sub("\\.2bit$", "",
                                             basename(assemblyTarget),
                                             ignore.case = TRUE), ".",
                                             sub("\\.2bit$", "",
                                             basename(assemblyQuery),
                                             ignore.case = TRUE),
                                    ".noClass.net")),
                   binaryChainNet="chainNet", binaryNetSyntenic="netSyntenic")

## End(Not run)
```

---

chainPreNet *chainPreNet*

Description

Wrapper function of chainPreNet: Remove chains that don't have a chance of being netted. This function doesn't work on Windows platform since Kent utilities only support Linux and Unix platform.

Usage

```
chainPreNet(allChain, assemblyTarget, assemblyQuery,
            allPreChain=paste0(sub("\\.2bit$", "", basename(assemblyTarget),
                                   ignore.case = TRUE), ".",
                               sub("\\.2bit$", "", basename(assemblyQuery),
                                   ignore.case = TRUE), ".all.pre.chain"),
            removeAllChain=TRUE, binary="chainPreNet")
```

Arguments

`allChain`	`character(1)`: file names of input *allChain* file.
`assemblyTarget`	`character(1)`: the file name of target assembly *twoBit* file.
`assemblyQuery`	`character(1)`: the file name of query assembly *twoBit* file.
`allPreChain`	`character(1)`: file names of merged *allPreChain* file.
`removeAllChain`	`boolean`: When `TRUE`, the input *allChain* file will be removed after the conversion.
`binary`	`character(1)`: the name/filename of the binary `chainPreNet` to call.

Value

`character(1)`: the file names of merged *allPreChain* file.

Author(s)

Ge Tan

References

http://hgdownload.cse.ucsc.edu/admin/exe/

See Also

`chainMergeSort`

Examples

```
## Not run:
## This example doesn't run because it requires two bit files and external
## Kent utilities.
  allChain <- file.path("/Users/gtan/OneDrive/Project/CSC/CNEr/axt",
                        "danRer10.hg38.all.chain")
  assemblyTarget <- "/Users/gtan/OneDrive/Project/CSC/CNEr/2bit/danRer10.2bit"
  assemblyQuery <- "/Users/gtan/OneDrive/Project/CSC/CNEr/2bit/hg38.2bit"
  chainPreNet(allChain, assemblyTarget, assemblyQuery,
              allPreChain=file.path(
                          "/Users/gtan/OneDrive/Project/CSC/CNEr/axt",
                                  paste0(sub("\\.2bit$", "",
                                             basename(assemblyTarget),
                                             ignore.case = TRUE), ".",
```

```
                                            sub("\\.2bit$", "",
                                                basename(assemblyQuery),
                                                ignore.case = TRUE),
                                                ".all.pre.chain")),
                removeAllChain=FALSE, binary="chainPreNet")

## End(Not run)
```

CNE-class *Class* "CNE"

Description

This class is used to store all intermediate and final results of CNE.

Usage

```
### Constructors:
CNE(assembly1=character(), assembly2=character(),thresholds=character(),
    CNE1=list(), CNE2=list(), CNEMerged=list(), CNERepeatsFiltered=list(),
    alignMethod=character())

### Accessor-like methods:
## S4 method for signature 'CNE'
assembly1(x)
## S4 method for signature 'CNE'
assembly2(x)
## S4 method for signature 'CNE'
thresholds(x)
## S4 method for signature 'CNE'
CNE1(x)
## S4 method for signature 'CNE'
CNE2(x)
## S4 method for signature 'CNE'
CNEMerged(x)
## S4 method for signature 'CNE'
CNERepeatsFiltered(x)

## ... and more (see Methods)
```

Arguments

assembly1	Object of class "character": The name of assembly1.
assembly2	Object of class "character": The name of assembly2.
thresholds	Object of class "character": The thresholds of CNE scan: window size and identity score in the form of "49_50".

CNE1	Object of class "list": The preliminary CNEs from axt file with assembly1 as reference.
CNE2	Object of class "list": The preliminart CNEs from axt file with assembly2 as reference.
CNEMerged	Object of class "list": The CNEs after merging CNE1 and CNE2.
CNERepeatsFiltered	Object of class "list": The CNEs after being realigned back to reference genome, with blat in current implementation.
alignMethod	Object of class "character": The method to realign CNEs back to reference genome.
x	Object of class "CNE": A "CNE" object.

Methods

assembly1 `signature(x = "CNE")`: Get the assembly1 name.

assembly2 `signature(x = "CNE")`: Get the assembly2 name.

CNE1 `signature(x = "CNE")`: Get the CNE1 results.

CNE2 `signature(x = "CNE")`: Get the CNE2 results.

CNEMerged `signature(x = "CNE")`: Get the merged CNE results.

CNERepeatsFiltered `signature(x = "CNE")`: Get the final CNE results.

thresholds `signature(x = "CNE")`: Get the thresholds used for scanning CNEs.

Author(s)

Ge Tan

Examples

```
showClass("CNE")
```

---

cneBlatedDanRer7Hg19 *The dataset cneBlatedDanRer7Hg19*

Description

This example dataset is the CNEs between hg19 and danRer7 after running blat program at the thresholds "45_50", "48_50" and "49_50".

Usage

```
data(cneBlatedDanRer7Hg19)
```

Examples

```
data(cneBlatedDanRer7Hg19)
```

CNEDanRer7Hg19 *CNEHg19DanRer7 and CNEHg19DanRer7 dataset*

Description

These two datasets are the direct output from ceScan.

Usage

```
data(CNEDanRer7Hg19)
```

Examples

```
data(CNEDanRer7Hg19)
```

CNEDensity-methods *CNEDensity function*

Description

This function queries the database and generates the CNEs density values.

Usage

```
CNEDensity(dbName, tableName, assembly1, assembly2, threshold,
           chr, start, end, windowSize, minLength=NULL)
```

Arguments

dbName	A object of character, the path of the local SQLite database.
tableName	A object of character, the name of table for this CNE data table. It can be missing when assembly1, assembly2 and threshold are provided.
assembly1	A object of character, the assembly to search.
assembly2	The comparison assembly. It can be missing when tableName is provided.
threshold	The threshold to search. It can be missing when tableName is provided.
chr	A object of character, the chromosome to query.
start, end	A object of integer, the start and end coordiante to fetch the CNEs.
windowSize	A object of integer, the window size in kb used to smooth the CNEs.
minLength	A object of integer, the minimal length of CNEs to fetch.

Value

A matrix is returned. The first column is the coordinates and the second column is the density values.

Methods

signature(tableName = "character", assembly1 = "character", assembly2 = "missing", threshold = "mis

signature(tableName = "missing", assembly1 = "character", assembly2 = "character", threshold = "cha

Author(s)

Ge Tan

Examples

```
dbName <- file.path(system.file("extdata", package="CNEr"),
                    "cne.sqlite")
chr <- "chr11"
start <- 31000000L
end <-  33000000L
windowSize <- 300L
minLength <- 50L
cneHg19DanRer7_45_50 <-
  CNEDensity(dbName=dbName,
             tableName="danRer7_hg19_45_50",
             assembly1="hg19", chr=chr, start=start,
             end=end, windowSize=windowSize,
             minLength=minLength)
cneHg19DanRer7_48_50 <-
  CNEDensity(dbName=dbName,
             tableName="danRer7_hg19_45_50",
             assembly1="hg19", chr=chr, start=start,
             end=end, windowSize=windowSize,
             minLength=minLength)
cneHg19DanRer7_49_50 <-
  CNEDensity(dbName=dbName,
             tableName="danRer7_hg19_45_50",
             assembly1="hg19", chr=chr, start=start,
             end=end, windowSize=windowSize,
             minLength=minLength)
```

---

cneHg19DanRer7_45_50 *These datasets of CNE density values.*

Description

These three datasets are output from CNEDensity.

Usage

```
data(cneHg19DanRer7_45_50)
```

Examples

```
data(cneHg19DanRer7_45_50)
data(cneHg19DanRer7_48_50)
data(cneHg19DanRer7_49_50)
```

---

cneMerge *CNE merge function*

Description

Remove the CNEs which overlap on both genomes.

Usage

```
cneMerge(cne1, cne2)
```

Arguments

cne1, cne2	A object of data.frame. The result from ceScan.

Value

A data.frame of CNEs is returned. In this table, the order of columns are consistent with cne1. For instance, if cne1 has the first three columns for zebrafish and next three columns for human, in the merged table, the first three columns are still the coordinates for zebrafish while the next three columns are coordinates for human.

Author(s)

Ge Tan

Examples

```
data(CNEHg19DanRer7)
data(CNEDanRer7Hg19)
cneMergedDanRer7Hg19 = mapply(cneMerge, CNEDanRer7Hg19, CNEHg19DanRer7,
                              SIMPLIFY=FALSE)
```

`fetchChromSizes` *fetchChromSizes function.*

Description

This function tries to automate the fetch of chrom sizes for assembly from UCSC and other sources.

Usage

```
fetchChromSizes(assembly)
```

Arguments

`assembly`	A `character` object: the canonical name of assembly, i.e., hg19 for UCSC.

Details

This function utilises mysql query for UCSC assemblies.

Value

A object of `Seqinfo` is returned.

Note

Currently the assemblies from UCSC are supported.

Author(s)

Ge Tan

Examples

```
fetchChromSizes("hg19")
fetchChromSizes("mm10")
```

finalCNE *finalCNE dataset*

Description

One example dataset in CNE class.

Usage

```
data(finalCNE)
```

Details

This is a subset of CNEs between hg19 and danRer7 on chromosome 11, from 31000000L to 32500000L based on hg19 coordinate.

Examples

```
data(finalCNE)
```

GRangePairs-class *GRangePairs objects*

Description

The GRangePairs class is a container for a pair of GRanges object that have same lengths.

Details

A GRangePairs object is a list-like object where each element describes a pair of genomic range. They do not necessarily have the same seqinfo, *i.e.*, the coordinates from the same assembly.

Constructor

GRangePairs(first=GRanges(), last=GRanges(), names=NULL): GRangePairs constructor.

Accessors

In the code snippets below, x is a GRangePairs object.

length(x): Return the number of granges pairs in x.

names(x), names(x) <- value: Get or set the names on x.

first(x), last(x): Get the "first" or "last" GRange for each grange pair in x. The result is a GRanges object of the same length as x.

seqnames(x): Get the seqname of first GRanges and last GRanges and return in a DataFrame object.

strand(x): Get the strand for each grange pair in x.

seqinfo(x): Get the information about the underlying sequences.

Vector methods

In the code snippets below, x is a GRangePairs object.

`x[i]`: Return a new GRangePairs object made of the selected genomic ranges pairs.

List methods

In the code snippets below, x is a GRangePairs object.

`x[[i]]`: Extract the i-th alignment pair as a GRangePairs object of length 2. As expected `x[[i]][1]` and `x[[i]][2]` are respectively the "first" and "last" granges in the pair.

`unlist(x, use.names=TRUE)`: Return the GRangePairs object conceptually defined by `c(x[[1]], x[[2]], ..., x[[len`
`use.names` determines whether x names should be propagated to the result or not.

Coercion

In the code snippets below, x is a GRangePairs object.

`grglist(x, use.mcols=FALSE)`:

Return a GRangesList object of length `length(x)` where the i-th element represents the ranges (with respect to the reference) of the i-th grange pair in x.

Note that this results in the ranges being *always* ordered consistently with the original "query template", that is, being in the order defined by walking the "query template" from the beginning to the end.

If `use.mcols` is TRUE and x has metadata columns on it (accessible with `mcols(x)`), they're propagated to the returned object.

`as(x, "GRangesList")`: Alternate ways of doing `grglist(x, use.mcols=TRUE)`.

`as(x, "GRanges")`: Equivalent of `unlist(x, use.names=TRUE)`.

Other methods

In the code snippets below, x is a GRangesList object.

`show(x)`: By default the `show` method displays 5 head and 5 tail elements. This can be changed by setting the global options `showHeadLines` and `showTailLines`. If the object length is less than (or equal to) the sum of these 2 options plus 1, then the full object is displayed.

Author(s)

Ge Tan

See Also

`Axt`

Examples

```
library(GenomicRanges)
first <- GRanges(seqnames=c("chr1", "chr1", "chr2", "chr3"),
                 ranges=IRanges(start=c(1, 20, 2, 3),
                                end=c(10, 25, 10, 10)),
                 strand="+")
last <- GRanges(seqnames=c("chr1", "chr10", "chr10", "chr20"),
                ranges=IRanges(start=c(1, 25, 50, 5),
                               end=c(8, 40, 55, 16)),
                strand="+")
namesGRangePairs <- c("a","b","c","d")
grangesPairs1 <- GRangePairs(first, last, names=namesGRangePairs)
grangesPairs2 <- GRangePairs(first, last)

## getters
names(grangesPairs1)
length(grangesPairs1)
first(grangesPairs1)
last(grangesPairs1)
seqnames(grangesPairs1)
strand(grangesPairs1)
seqinfo(grangesPairs1)

## setters
names(grangesPairs2) <- namesGRangePairs

## Vector methods
grangesPairs1[1]

## List methods
grangesPairs1[[1]]
unlist(grangesPairs1)

## Coersion
grglist(grangesPairs1)
as(grangesPairs1, "GRangesList")
as(grangesPairs1, "GRanges")
as(grangesPairs1, "DataFrame")
as.data.frame(grangesPairs1)

## Combining
c(grangesPairs1, grangesPairs2)
```

`lastal` *lastal wrapper*

Description

Wrapper function of `lastal` to do the pairwise whole genome alignment. This function doesn't work on Windows platform.

Usage

```
lastal(db, queryFn,
       outputFn=sub("\\.(fa|fasta)$", ".maf",
                    paste(basename(db), basename(queryFn), sep = ","),
                    ignore.case = TRUE),
       distance=c("far", "medium", "close"), binary="lastal",
       mc.cores=getOption("mc.cores", 2L), echoCommand=FALSE)
```

Arguments

db	character(1): the file name of target assembly's lastal index.
queryFn	character(1): the file name of query assembly *fasta* file.
outputFn	character(1): the file name of the output *maf* file.
distance	It can be "far", "medium" or "close". It decides the score matrix used in *lastz* aligner. See '?scoringMatrix' for more details.
binary	character(1): the name/filename of the binary lastal to call.
mc.cores	integer(1): the number of threads to use. By default, getOption("mc.cores", 2L).
echoCommand	boolean(1): When TRUE, only the command to run lastal is returned.

Value

A character(1) vector of ouput *maf* file names.

Note

lastal aligner must be installed on the machine to use this function.

Author(s)

Ge Tan

References

http://last.cbrc.jp/

See Also

lastz

Examples

```
## Not run:
  assemblyDir <- "/Users/gtan/OneDrive/Project/CSC/CNEr/2bit"
  ## Build the lastdb index
  system2(command="lastdb", args=c("-c", file.path(assemblyDir, "danRer10"),
                                   file.path(assemblyDir, "danRer10.fa")))

  ## Run lastal aligner
```

```
    lastal(db=file.path(assemblyDir, "danRer10"),
         queryFn=file.path(assemblyDir, "hg38.fa"),
         outputFn=file.path(axtDir, "danRer10.hg38.maf"),
         distance="far", binary="lastal", mc.cores=4L)

    ## maf to psl
    psls <- file.path(axtDir, "danRer10.hg38.psl")
    system2(command="maf-convert",
            args=c("psl", file.path(axtDir, "danRer10.hg38.maf"),
                   ">", psls))

## End(Not run)
```

`lastz` *lastz wrapper*

Description

Wrapper function of `lastz` to do the pairwise whole genome alignment. This function doesn't work on Windows platform.

Usage

```
lastz(assemblyTarget, assemblyQuery, outputDir = ".",
      chrsTarget = NULL, chrsQuery = NULL,
      distance = c("far", "medium", "close"), binary = "lastz",
      mc.cores = getOption("mc.cores", 2L), echoCommand = FALSE)
```

Arguments

`assemblyTarget`	`character(1)`: the file name of target assembly *twoBit* file.
`assemblyQuery`	`character(1)`: the file name of query assembly *twoBit* file.
`outputDir`	`character(1)`: the folder to put the generated *lav* files.
`chrsTarget`	NULL or `character(n)`: when it's NULL, all the available chromosomes from the target assembly will be aligned.
`chrsQuery`	NULL or `character(n)`: when it's NULL, all the available chromosomes from the query assembly will be aligned.
`distance`	It can be "far", "medium" or "close". It decides the score matrix used in *lastz* aligner. See '?scoringMatrix' for more details.
`binary`	`character(1)`: the name/filename of the binary `lastz` to call.
`mc.cores`	`integer(1)`: the number of threads to use. By default, `getOption("mc.cores", 2L)`.
`echoCommand`	`boolean(1)`: When `TRUE`, only the command to run `lastz` is returned.

Value

A `character(n)` vector of ouput *lav* file names.

Note

`lastz` aligner must be installed on the machine to use this function.

Author(s)

Ge Tan

References

http://www.bx.psu.edu/~rsharris/lastz/

See Also

lavToPsl

Examples

```
  ## Not run:
  ## This example doesn't run because it requires two bit files and external
  ## Kent utilities.
    assemblyTarget <- "/Users/gtan/OneDrive/Project/CSC/CNEr/2bit/danRer10.2bit"
    assemblyQuery <- "/Users/gtan/OneDrive/Project/CSC/CNEr/2bit/hg38.2bit"
    lavs <- lastz(assemblyTarget, assemblyQuery,
                  outputDir="/Users/gtan/OneDrive/Project/CSC/CNEr/axt",
            chrsTarget=c("chr1", "chr2", "chr3"),
            chrsQuery=c("chr1", "chr2", "chr3"),
            distance="far", mc.cores=4)

## End(Not run)
```

`lavToPsl` *lavToPsl*

Description

Wrapper function of `lavToPsl`: Convert blastz lav to psl format. This function doesn't work on Windows platform since Kent utilities only support Linux and Unix platform.

Usage

```
lavToPsl(lavs, psls=sub("\\.lav$", ".psl", lavs, ignore.case = TRUE),
         removeLav=TRUE, binary="lavToPsl")
```

Arguments

`lavs`	`character`(n): file names of input *lav* files.
`psls`	codecharacter(n): file names of output *psl* files. By default, in the same folder of input lav files with same names.
`removeLav`	`boolean`: When `TRUE`, the input *lavs* files will be removed after the conversion.
`binary`	`character`(1): the name/filename of the binary `lavToPsl` to call.

Value

character(n): the file names of output *psl* files.

Author(s)

Ge Tan

References

http://hgdownload.cse.ucsc.edu/admin/exe/

See Also

lastz

Examples

```
  ## Not run:
  ## This example doesn't run because it requires lav files from previous steps
  ## and external Kent utilities.
    lavs <- tools::list_files_with_exts(
              dir="/Users/gtan/OneDrive/Project/CSC/CNEr/axt", exts="lav")
    lavToPsl(lavs, removeLav=FALSE, binary="lavToPsl")

## End(Not run)
```

---

makeGRBs *makeGRBs*

Description

Make Genomic Regulatory Blocks (GRBs) boundaries prediction from a set of CNEs.

Usage

```
  makeGRBs(x, winSize=NULL, genes=NULL, ratio=0.5)
```

Arguments

x	GRangesList object of a set of CNEs to use.
winSize	integer: the smoothing window size for CNE densities in kb. This value depends on the genome size of the reference genome. A larger genome requires bigger windows size. For instance, 300kb is the appropriate windows size for human genome. By default, it is determined internally based on the genome size.
genes	NULL or GRanges object: the protein-coding genes ranges.
ratio	numeric(1) between 0 and 1: the threshold to control the stringency of the GRBs. Higher value, shorter and fewer GRBs, and vice versa.

Details

First we calculated the CNE densities from the CNEs. Then we segment the regions according to the value of CNE densities. The regions with CNE densities above the expected CNE densities * ratio are consider as putative GRBs. As last step, the putative GRBs that do not encompass any gene are filtered out.

Value

A GRanges object with GRB coordinates is returned.

Author(s)

Ge Tan

Examples

```
  ## Not run:
    ## Add example CNEs to make an example.

## End(Not run)
```

---

mismatchSummary *Utility functions related to Axt alignment*

Description

A collection of different functions used to deal with Axt object.

Usage

```
mismatchSummary(x, ...) ## mismatch number and proportion
```

Arguments

x	An Axt object
...	Currently not used.

Details

'mismatchSummary': a numeric vector giving the numner of mismatches and the proportion of mismatches.

Author(s)

Ge Tan

Examples

```
axtFilesHg19DanRer7 <- file.path(system.file("extdata", package="CNEr"),
                                 "hg19.danRer7.net.axt")
axtHg19DanRer7 <- readAxt(axtFilesHg19DanRer7)
mismatchSummary(axtHg19DanRer7)
```

N50 *Assembly statistics.*

Description

Calculate the N50, N90 values for a fasta or 2bit file.

Usage

```
N50(filepath)
N90(filepath)
```

Arguments

filepath	The path name of a fasta or 2bit file.

Details

This function calculates the N50, N90 values for an assembly. The N50 value is calculated by first ordering every contig/scaffold by length from longest to shortest. Next, starting from the longest contig/scaffold, the lengths of each contig are summed, until this running sum equals one-half of the total length of all contigs/scaffolds in the assembly. Then the length of shortest contig/scaffold in this list is the N50 value. Similar procedure is used for N90 but including 90% of the assembly.

Value

An integer value of N50 or N90 value.

Author(s)

Ge Tan

netToAxt *netToAxt*

Description

Wrapper function of `netToAxt` and `axtSort`: convert net (and chain) to axt, and sort axt files. This function doesn't work on Windows platform since Kent utilities only support Linux and Unix platform.

Usage

```
netToAxt(in.net, in.chain, assemblyTarget, assemblyQuery,
         axtFile=paste0(sub("\\.2bit$", "", basename(assemblyTarget),
                        ignore.case = TRUE), ".",
                    sub("\\.2bit$", "", basename(assemblyQuery),
                        ignore.case = TRUE), ".net.axt"),
         removeFiles=FALSE,
         binaryNetToAxt="netToAxt", binaryAxtSort="axtSort")
```

Arguments

`in.net`	`character(1)`: file names of input *net* file.
`in.chain`	`character(1)`: file names of input *chain* file.
`assemblyTarget`	`character(1)`: the file name of target assembly *twoBit* file.
`assemblyQuery`	`character(1)`: the file name of query assembly *twoBit* file.
`axtFile`	`character(1)`: file names of output *axt* file.
`removeFiles`	`boolean`: When `TRUE`, the input *net* and *chain* files will be removed after the conversion.
`binaryNetToAxt`	`character(1)`: the name/filename of the binary `netToAxt` to call.
`binaryAxtSort`	`character(1)`: the name/filename of the binary `axtSort` to call.

Value

`character(1)`: the file name of output *axt* file.

Author(s)

Ge Tan

References

http://hgdownload.cse.ucsc.edu/admin/exe/

See Also

chainNetSyntenic

Examples

```
## Not run:
  ## This example doesn't run because it requires two bit files and external
  ## Kent utilities.
  in.net <- file.path("/Users/gtan/OneDrive/Project/CSC/CNEr/axt",
                      "danRer10.hg38.noClass.net")
  in.chain <- file.path("/Users/gtan/OneDrive/Project/CSC/CNEr/axt",
                        "danRer10.hg38.all.pre.chain")
  assemblyTarget <- "/Users/gtan/OneDrive/Project/CSC/CNEr/2bit/danRer10.2bit"
  assemblyQuery <- "/Users/gtan/OneDrive/Project/CSC/CNEr/2bit/hg38.2bit"
  netToAxt(in.net, in.chain, assemblyTarget, assemblyQuery,
           axtFile=file.path("/Users/gtan/OneDrive/Project/CSC/CNEr/axt",
                             paste0(sub("\\.2bit$", "",
                                        basename(assemblyTarget),
                                        ignore.case = TRUE), ".",
                                    sub("\\.2bit$", "",
                                        basename(assemblyQuery),
                                        ignore.case = TRUE),
                                    ".net.axt")),
           removeFiles=FALSE,
           binaryNetToAxt="netToAxt", binaryAxtSort="axtSort")

## End(Not run)
```

---

qSizesDanRer7 *The chromosome sizes data.*

Description

The chromosome sizes data of hg19 and danRer7.

Usage

```
data(qSizesDanRer7)
       data(qSizesHg19)
```

Source

http://hgdownload.soe.ucsc.edu/downloads.html

Examples

```
data(qSizesDanRer7)
data(qSizesHg19)
```

queryCNEData *Query the CNEData package to fetch the CNEs*

Description

Query the CNEData package to fetch the CNEs based on target, query species, winSize and identity.

Usage

```
queryCNEData(dbName, target, query, winSize, identity,
             type=c("target", "all"))
```

Arguments

dbName	The path of SQLite database.
target, query	The CNEs between target and query species.
winSize, identity	The thresholds of CNEs to fetch on identity over winSize.
type	Which set of CNEs are returned. When it is "all", the CNEs of target always on the left side of returned data.frame.

Value

A data.frame of CNEs coordinates in chr, start, end.

Author(s)

Ge Tan

---

readAxt *readAxt*

Description

This function reads the *axt* files into a Axt object.

Usage

```
readAxt(axtFiles)
```

Arguments

axtFiles	character(n): file names of the *axt* files to read.

Details

This function reads the *axt* files of two assemblies. It can be a single big *axt* file or several small *axt* files. Different from the start coordinate in *axt* file, the start coordinate in `Axt` object is 1-based.

Value

A object `Axt` is returned.

Author(s)

Ge Tan

See Also

`Axt`

Examples

```
axtFilesHg19DanRer7 <- file.path(system.file("extdata", package="CNEr"),
                                 "hg19.danRer7.net.axt")
axtHg19DanRer7 <- readAxt(axtFilesHg19DanRer7)
```

`readBed` *readBed*

Description

Read the coordiantes information from a bed file.

Usage

```
readBed(bedFile)
```

Arguments

`bedFile`	The `character`(1) file name of the 'bed' file to read.

Details

This function is designed to read the bed file for the first three columns, *i.e.*, "chrom", "chromStart", "chromEnd". The strand information is also stored when available.

In bed file, the "chromStart" is on 0-based coordinate while "chromEnd" is on 1-based coordinate. For example, the first 100 bases of a chromosome are defined as "chromStart"=0, "chromEnd"=100, and span the bases numbered 0-99. When it is read into GRanges, both the chromStart and chromEnd are on 1-based coordinate, *i.e.*, "chromStart"=1 and "chromEnd"=100.

Value

A GRanges is returned. When no strand information is availabe in bed file, all the ranges are assumed to be on the positive strand.

Author(s)

Ge Tan

References

https://genome.ucsc.edu/FAQ/FAQformat.html#format1

See Also

import.bed

Examples

```
bedHg19Fn <- file.path(system.file("extdata", package="CNEr"),
                       "filter_regions.hg19.bed")
bedHg19 <- readBed(bedHg19Fn)
```

---

readCNERangesFromSQLite

readCNERangesFromSQLite function

Description

Query the SQLite database based on chromosome, coordiantes and some other criterias. Usually not to be used directly. For the CNE density plot, fetchCNEDensity function should be used.

Usage

```
readCNERangesFromSQLite(dbName, tableName, chr, start, end,
                        whichAssembly=c("L","R"), minLength=NULL)
```

Arguments

dbName	A object of character, the path of the local SQLite database.
tableName	A object of character, the name of table for this CNE data table.
chr	A object of character, the chromosome to query
start, end	A object of integer, the start and end coordiante to fetch the CNEs.
whichAssembly	A object of character, the genome to fetch is in the "Left" columns or "Right" columns of the table.
minLength	A object of integer, the minimal length for selected CNEs.

Value

A object of IRanges is retu

Author(s)

Ge Tan

Examples

```
dbName <- file.path(system.file("extdata", package="CNEr"),
                    "cne.sqlite")
chr <- "chr11"
start <- 31000000L
end <-  33000000L
minLength <- 50L
tableName <- "danRer7_hg19_45_50"
fetchedCNERanges <- readCNERangesFromSQLite(dbName, tableName, chr,
                                            start, end, whichAssembly="L",
                                            minLength=minLength)
```

---

reverseCigar *reverseCigar function*

Description

This function reverses the cigar string, i.e., 20M15I10D will be reversed to 10D15I20M.

Usage

```
reverseCigar(cigar, ops=CIGAR_OPS)
```

Arguments

cigar	A character vector of cigar strings.
ops	A character vector of the extended CIGAR operations. By default, CIGAR_OPS is used.

Value

A character vector contains the revered cigar strings.

Author(s)

Ge Tan

See Also

cigar-utils

Examples

```
cigar = c("20M15I10D", "10D15I20M")
reverseCigar(cigar)
```

---

saveCNEToSQLite-methods

saveCNEToSQLite function

Description

This function save the CNE results into a local SQLite database.

Usage

```
saveCNEToSQLite(CNE, dbName, tableName, overwrite=FALSE)
```

Arguments

CNE	An object of data.frame, the CNE data table or an object of CNE.
dbName	An object of character, the path of the local SQLite database.
tableName	An object of character, the name of table for this CNE data table, or missing when CNE is an object of CNE.
overwrite	An object of boolean, whether or not to overwrite the table with same table name.

Details

The input CNE table should have the colnames "chr1", "start1", "end1", "chr2", "start2", "end2", "strand", "similarity", "cigar". After the bin indexing, two additional columns "bin1" and "bin2" will be added before the column "chr1" and "chr2", respectively.

If the input CNE is a CNE object, the tableName will be a combination of assembly names and thresholds. For instance, "danRer7_hg19_49_50" for "hg19" and "danRer7" with threshold "49_50".

Author(s)

Ge Tan

Examples

```
dbName = tempfile()
data(cneBlatedDanRer7Hg19)
for(i in 1:length(cneBlatedDanRer7Hg19)){
  tableName = paste("danRer7_hg19", names(cneBlatedDanRer7Hg19)[i],
                    sep="_")
  saveCNEToSQLite(cneBlatedDanRer7Hg19[[i]], dbName, tableName,
                  overwrite=TRUE)
}
```

```
data(finalCNE)
saveCNEToSQLite(finalCNE, dbName=dbName, overwrite=TRUE)
```

---

scoringMatrix *scoringMatrix*

Description

Generate the scoring matrix for *lastz* aligner.

Usage

```
scoringMatrix(distance = c("far", "medium", "close"))
```

Arguments

distance	It can be "far", "medium" or "close". It decides the score matrix used in *lastz* aligner. Generally, if two species are close to each other at human and chimp level, "close" should be used. If two species have a divergent time of 100 MYA, "far" should be used. In other cases, use "medium".

Value

A `matrix` of the scoring matrix is returned.

Note

HOXD70 is medium. HoxD55 is far. human-chimp.v2 is close.

Author(s)

Ge Tan

References

http://genomewiki.ucsc.edu/index.php/Hg38_17-way_conservation_lastz_parameters

See Also

lastz

Examples

```
scoringMatrix(distance="far")
```

subAxt-methods subAxt *method*

Description

Get subset of Axt alignments based on chromosome and ranges.

Usage

```
subAxt(x, chr, start, end, select=c("target", "query"), qSize=NULL)
```

Arguments

x	A object of Axt.
chr	A object of character. The chromosome name to extract.
start, end	A object of integer. These ranges should be based on the positive strand. When select is "query", the reverse complement alignments which lay inside this range will also be selected.
select	When select is "target", the subset criteria is for target alignments in axts. When select is "query", the subset criteria is for query alignments in axts.
qSize	When select is "query", qSize must be provided and is the length of chromosome chr.

Details

Usually when we want to subset some axts from a Axt object, we care about all the axts within certain range. The axts can come from the axt file with chr as reference (i.e., target sequence), or the axt file with chr as query sequence. When the chr is query sequence, it can be on the negative strand. Hence, the size of chromosome is necessary to convert the search range to a range on negative strand coordinate.

When one axt is partially overlapped with the range, subset of the axt will be extract. If the extracted axt alignment has gaps at the beginning or the end, the gap columns will be chopped. Therefore, the coordinate of alignments will be changed accordingly.

Value

A subset of Axt object is returned.

Author(s)

Ge Tan

Examples

```
axtFilesHg19DanRer7 <- file.path(system.file("extdata", package="CNEr"),
                                 "hg19.danRer7.net.axt")
axtHg19DanRer7 <- readAxt(axtFilesHg19DanRer7)
subAxt(axtHg19DanRer7, chr="chr11", start=31500000, end=32500000,
       select="target")
subAxt(axtHg19DanRer7, chr="chr11", start=c(31082021, 32461267),
       end=c(31082862,32461581), select="target")
```

`writeAxt` `writeAxt` *function*

Description

Write an axt object into file.

Usage

```
writeAxt(axt, con)
```

Arguments

`axt`	A `Axt` object to write.
`con`	A connection object or a character string.

Author(s)

Ge Tan

See Also

`readAxt`

Examples

```
axtFilesHg19DanRer7 <- file.path(system.file("extdata", package="CNEr"),
                                 "hg19.danRer7.net.axt")
axtHg19DanRer7 <- readAxt(axtFilesHg19DanRer7)
writeAxt(axtHg19DanRer7, con=tempfile())
```

Index