Package 'CNEr'

October 7, 2014

Version 1.0.0

Date 2014-03-12

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(>= 2.26.3), RSQLite(>= 0.11.4), GenomicRanges(>= 1.12.5), rtracklayer(>= 1.20.4), XVector(>= 0.2.0), DBI(>= 0.2-7), GenomicAlignments(>= 0.99.17), methods, IRanges(>= 1.19.38)

Depends R (>= 3.0.2)

Suggests Gviz(>= 1.7.4), RUnit, BiocGenerics

LinkingTo IRanges, XVector

System Requirements blat

License GPL-2 | file LICENSE

License_restricts_use yes

URL http://ancora.genereg.net/

Type Package

biocViews GeneRegulation, Visualization, DataImport

NeedsCompilation yes

LazyData no

Collate AllGenerics.R AllClasses.R utils.R ceScan.R plot.R makeGeneDbFromUCSC.R io.R scoringMatrix.R subAxt-methods.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(), symCount=integer())

## 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")
```

---

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	Object of character. The length can be one or more.

Value

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

Author(s)

Ge Tan

See Also

link{readAxt}

Examples

```
axtFilesHg19DanRer7 =
  file.path(system.file("extdata", package="CNEr"),
            "hg19.danRer7.net.axt")
  ans = 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

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)
```

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)
```

readAxt *readAxt function.*

Description

This function reads the axt files into a Axt object.

Usage

```
readAxt(axtFiles)
```

Arguments

axtFiles	Object of character. The length can be one or more.

Details

This function reads the axt files. The coordinates 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 function*

Description

readBed reads bed file with an embeded C IO function.

Usage

```
readBed(bedFile)
```

Arguments

bedFile	The path of bed file.

Details

This function is designed to read the bed file only with three mandatory columns, i.e., "chrom", "chromStart", "chromEnd". This function utilises the C interface for speedy import. For general bed file import, please use the `import.bed` from package `rtracklayer`.

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

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)
```

---

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 be written.
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