

The **CNEr** package overview

Ge Tan

May 2, 2014

Contents

1	Introduction	1
2	Pipeline of the package	2
2.1	CNE identification	2
2.2	CNE visualisation	2
3	Input	2
3.1	axt file	3
3.2	bed files	5
3.3	Chromosome size of query assembly	6
4	CNE identification	7
4.1	net alignments scan	8
4.2	Merge CNEs	9
4.3	Realignment of CNEs	10
5	CNE storage and query	11
5.1	CNE storage and query	11
6	CNEs visualisation	12
6.1	Gene annotation visualisation	12
6.2	CNEs horizon plot	14
7	Conclusion	17

1 Introduction

Conserved noncoding elements (CNEs) are pervasive class of elements clustering around genes with roles in development and differentiation in Metazoa (Woolfe *et al.*, 2004). While many have been shown to act as long-range developmental enhancers (Sandelin *et al.*, 2004), the source of their extreme conservation remains unexplained. To study the evolutionary dynamics of these elements and their relationship to the genes around which they cluster, it is essential to be

able to produce genome-wide sets of these elements for a large number of species comparisons, each with multiple size and conservation thresholds.

This *CNEr* package aims to detect CNEs and visualise them along the genome. For performance reasons, the implementation of CNEs detection and corresponding I/O functions are primarily written as C extensions to R. We have used *CNEr* to produce sets of CNEs by scanning pairwise whole-genome net alignments with multiple reference species, each with two different window sizes and a range of minimum identity thresholds. Then, to pinpoint the boundaries of CNE arrays, we compute the CNE densities as the percentages of length covered by CNEs within a user specified window size. Finally, we describe a novel visualisation method using horizon plot tracks that shows a superior dynamic range to the standard density plots, simultaneously revealing CNE clusters characterized by vastly different levels of sequence conservation. Such CNE density plots generated using precise locations of CNEs can be used to identify genes involved in developmental regulation, even for novel genes that are not annotated yet.

2 Pipeline of the package

This section will briefly demonstrate the pipeline of CNE identification and visualisation. More detailed usage of each step will be described in following sections with a concrete example of CNE identification and visualisation for Human (hg19) and Zebrafish (danRer7).

2.1 CNE identification

1. axtNets: The axtNet files can be downloaded from UCSC or generated by itself
2. scan alignments: The regions with minimal **I** identities over **C** columns.
3. remove elements: Elements that overlap with annotated exons/repeats.
4. merge elements to get CNEs: Elements that overlap on both genomes.

2.2 CNE visualisation

1. display parameters: Chromosome, start, end, smooth window size.
2. horizon plot: Visualise CNE densities.

3 Input

CNEr starts from [axt](#) net files of two genomes pairwise alignments and bed files for filtering. UCSC already provides a set of precomputed axt files on <http://hgdownload.soe.ucsc.edu/downloads.html> for most of popular genomes.

In case the axt files are not available from UCSC, you can always generate the axt net files by following the UCSC wiki http://genomewiki.ucsc.edu/index.php/Whole_genome_alignment_howto.

3.1 axt file

So far, there is no suitable class to store the axt files in Bioconductor. Hence we created a **S4** class *Axt* to hold the content from axt files. Basically, it utilises **GRanges** from *GenomicRanges* package and **DNAStringSet** from *Biostrings* package.

To read axt file into R, *CNEr* provides `readAxt` function for highly efficient reading, which heavily depends on Kent's utilities source code (Kent *et al.*, 2002). The axt file can be gzipped or in plain text file. The alignments between two genomes can also be in one file or in several files, such as chr1.hg19.mm10.net.axt.gz, chr2.hg19.mm10.net.axt.gz, etc.

```
> library(CNEr)
> axtFilesHg19DanRer7 <- file.path(system.file("extdata", package="CNEr"),
+                                 "hg19.danRer7.net.axt")
> axtFilesDanRer7Hg19 <- file.path(system.file("extdata", package="CNEr"),
+                                 "danRer7.hg19.net.axt")

> axtHg19DanRer7 <- readAxt(axtFilesHg19DanRer7)
> axtDanRer7Hg19 <- readAxt(axtFilesDanRer7Hg19)

> axtHg19DanRer7

A Axt with 133 alignment pairs:
  1 chr11 31082021 31082458 chr25 15030563 15031009 +   246
ATTTTATGTGCAAGATAGACTTTAAATTTAAAT...TATTAAAGCACATTTTAATTGAAAAAAGTCAA
ATTTAAAGTGATAGTTTGCCCCAAAATTTAATT...AATAAAACCATAATTCAACTAACAGACATTAAA
  2 chr11 31082562 31082862 chr25 15036393 15036688 +   422
GGGAAGGAAGGGCTGGTTAGGAGTACAGTAGCTA...TCCTTTGGATAAACTAATGTAAACAAACACATT
GGAAAAGGGGGGC-----GAGGACGAGAGC--...CTCATCTGATTGGCTTGTAGCAACAACCTCCATT
  3 chr11 31085908 31086035 chr7 60364309 60364442 -   5679
GGGCTTTTGAGATATTTCAATTATCTTCAAGAAA...GTCAGTACCCTACCTCTGAGCAGAAGCCTTAGG
GGGCTCTGGAGGTATTTTCAGAAAAATCAGAGAAA...GTCAACGTGGAACCTTAGAGCGAGAGCCTAGGG
  4 chr11 31087045 31087145 chr25 15037012 15037110 +   1743
TTGTGTAGTTGCTGATTGTTTGAAGTATTCAA...AATGAAAGAGCTGCATTAAGCATAGATAAGATA
TTAAATAATTGCTTGATTATTTTGGACTGTTCA...A--GAAAGCTTAATGTTATTGTTACATAAATTA
  5 chr11 31090791 31090882 chr7 60364679 60364769 -   1556
TATTCAGATAATTTTGAACGAGATTTTGAATC...AAAAGAAGTTTAATCCAAAAAATTAGGATTT
TATTTTAATGATTGTAATA-GTCATTTTGAAG...AAAACAAAATGAATGTATAATACAATATAATTT
... ..
129 chr11 32449499 32449619 chr25 15559655 15559778 +   6263
TACCTGCTGTAGGGCGTCCTCAGCAGCAAAGCCT...AGTACTGCTGCTCACCTG---CAGAGAGAACCG
TACCTGTTGTATGGGTTTCTCAGCAGCAGAGCTT...GATACTGCTGATCCCCTGAAACAAACAGAACAG
```

```

130 chr11 32450037 32450174 chr25 15564924 15565061 + 6952
GCTTACCCAGCGAGCCCTGCTGGCCCATGGGATC...CGTCGAAGGTGACCGTGCTGTAACCTGCGGGAG
GTTTACCCATGTTGCTCTGCTGTGTGATTGAGTC...CGTCAAAGCAACCGTGCCGTAACCTATAAAAG
131 chr11 32456239 32456689 chr25 15570666 15571071 + 13604
TACTTACCCTGATTGCGAATAGCGGGCTGGCTCT...GCGCGTTCAGGTCCCGCACGTCGGAGCCCATT
TCCTCACCTTGATTCTGAGCTGGATTGGCTGT...GAGTGTGAGGTCACGAACATCAGAACCCATT
132 chr11 32456981 32457152 chr25 15571250 15571407 + 1488
GAGTAGGTGGGAGGGAGGGCGGGAAGTGGGGGAG...GGGCCGGCGCGCGGGGAAGAGGAGGAGCCAGG
GAATTGGAGAAAAGAAGAACAGGA-----GGAT...CAGTCCGAGGAGTCGGGAGGAGGAGGAGTCAGG
133 chr11 32461267 32461582 chr25 15574876 15575192 + 6228
GTCCCGAGCTCCCGAGAGTCCTCAGTGAAAGG...AACACAGGGGAAAAAATGGGGAAAAAGAAAAA
GTCACGTGATCTCAGAAGGTC-----AGGA...ACTCGGGGAAGAGGAGAGCGAGGAAAAAGAAAAA
> axtDanRer7Hg19

```

```

A Axt with 190 alignment pairs:
 1 chr12 46824249 46824342 chr11 31823296 31823389 + -311
GGAA---CACGTGTAAATCTG-----ATTTTCA...CAGATGCGTTGTAACAAAGCCATGAAAAATGTTT
AGAATTTTACTCACACATCCGTTGGACACCTGCA...-----TTTCAAGACAAAAATAAAATTGTTT
 2 chr18 41195293 41195862 chr11 103744646 103745211 - 3029
CATCATATTACAGTGCAAAATAACATAAAATATA...TAAATTGCATC--ATATTGAAATACAAAATAAA
CATAATAATACAATG--AAAGAAAGAAAAATCCA...TATATTTTATTAATATTAATTAATGAACAAAA
 3 chr18 44008076 44008197 chr11 32410604 32410725 + 7229
TCAGACGGTGGGCTGCAGTTTGCTCATGTTTCTC...AGCTCGGCCAGCGGCACGTAAACGGCTTCTCAC
TCAAAGCGCCAGCTGGAGTTTGGTCATGTTTCTC...AAGTTGGCCACCGACAGCTGAAGGGCTTTTCAC
 4 chr18 44008198 44008391 chr11 32413401 32413612 + 6896
CTGAGACAACACACACACACATTTTAATCAAG...TGACACGTCTCACACTGAAAGGGTTTGACGCCT
CTGAGTCTAAAC---CTTAGAACTTTTACACTAG...TGACAAGTTTACACTGGAATGGTTTCACACCT
 5 chr18 44009282 44009389 chr11 32414210 32414318 + 5405
ACCTGTGTGTCTGCGCTGGTGTCTCTTCAGCTGG...TGATACGGCTTTTC-CCCTGAACACCACACACA
ACCTGTATGTCTCCTTTGGTGTCTTTTGAGCTGG...TGGTATGGTTTCTCACCTTGGGGAAGACACATA
... ..
186 chr7 16881835 16881957 chr11 103785288 103785409 - 6493
CTTTTTCACAAGCCATTAGCTCCTGGAGCAACAT...AAGTCAGAGAAGTTGCTCTGAGCTGCAGGGAGG
CTTTTTCACAATCCATTAATTCCTGGATCAACAT...AAGTCAGAGACATT-CTCTGAGCAGCAGGGAAG
187 chr7 16911295 16911385 chr11 103915635 103915726 - 1556
AAATTATATTGTATTATACATTCAATTTTGTTTT...TTTCAAAAATGAC-TATTACAATCATTAATAAATA
AAATCCTAATTTTTTTGGATTAAACTTCTTTT...ATTCAAAAATCTCGTTCCAAAATTATCTGAATA
188 chr7 16911622 16911755 chr11 103920482 103920609 - 5679
CCCTAGGCTCTCGCTCTAAGGTTCCACGTTGACA...TTCTCTGATTTTCTGAAATACCTCCAGAGCCC
CCTAAGGCTTCTGCTCAGAGGTAGGGTACTGACA...TTCTTGAAGATAATTGAAATATCTCAAAAGCCC
189 chr7 16918825 16918945 chr11 103923739 103923855 - 5401
CCAGCGCTGCGCACTGACCTGGCT-GTCGCTCAC...GACAAGCCGTCCCAGCCGCAGCACATCTGTGT
CCAACAAAGCATGTGACCCAGCTTGTCAACCCAG...GACAGTTTGCCCTTTG-----GGCACATCTGTGT
190 chr7 27639642 27639730 chr11 102899904 102899999 - 642
CTTTTCTTTTAATTGAATTATATT---TAAGATT...TTAAA--AGTTAAGATCAAGTTCCCAGAGTCTCT
ACTTTCTTTTGTAGATATTCATTCTGTAGGTC...AAAGATCACTTAAGAAAGAGGTCCCCCAGTCTT

```

3.2 bed files

The gene annotation information, including exons coordinates, repeats, is provided in a bed file. Here we summarise a table of filtering information we used:

Assembly	Name	Exon	Repeat
hg19	Human	RefSeq Genes, Ensembl Genes, UCSC Known Genes	RepeatMasker
mm10	Mouse	RefSeq Genes, Ensembl Genes, UCSC Known Genes	RepeatMasker
xenTro3	Frog	RefSeq Genes, Ensembl Genes	RepeatMasker
tetNig2	Tetraodon	Ensembl Genes	
canFam3	Dog	RefSeq Genes, Ensembl Genes	RepeatMasker
galGal4	Chicken	RefSeq Genes, Ensembl Genes	RepeatMasker
danRer7	Zebrafish	RefSeq Genes, Ensembl Genes	RepeatMasker
fr3	Fugu	RefSeq Genes	RepeatMasker
anoCar2	Lizard	Ensembl Genes	RepeatMasker
equCab2	Horse	RefSeq Genes, Ensembl Genes	RepeatMasker
oryLat2	Medaka	RefSeq Genes, Ensembl Genes	RepeatMasker
monDom5	Opossum	RefSeq Genes, Ensembl Genes	RepeatMasker
gasAcu1	Stickleback	RefSeq Genes, Ensembl Genes	RepeatMasker
rn5	Rat	RefSeq Genes, Ensembl Genes	RepeatMasker
dm3	D. melanogaster	RefSeq Genes, Ensembl Genes	RepeatMasker
droAna2	D. ananassae		RepeatMasker
dp3	D. pseudoobscura		RepeatMasker
ce4	C. elegans	RefSeq Genes	RepeatMasker
cb3	C. briggsae		RepeatMasker
caeRem2	C. remanei		RepeatMasker
caePb1	C. brenneri		RepeatMasker

Of course, more customised gene annotation can be included in the bed file. To import bed file into **GRanges** in RR, *rtracklayer* provides a general function `import.bed` to do that. However, if your bed file only contains 3 columns: chromosome, start and end, that are the information needed by *CNEr*, then it is better to use the `readBed` function that is much faster with C implementation.

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

```
Reading /private/var/folders/1c/_5ks0mjd3gvf_dq14g3thsr40000gn/T/Rtmp83Fap1/Rinst7c9f15bed9b
```

```
> bedHg19
```

```
GRanges with 5574 ranges and 0 metadata columns:
```

```
      seqnames      ranges strand
   <Rle>          <IRanges>  <Rle>
[1]   chr11 [30000001, 30000106]    +
[2]   chr11 [30000126, 30000274]    +
```

```

[3] chr11 [30001660, 30003939] +
[4] chr11 [30003969, 30004060] +
[5] chr11 [30004481, 30004844] +
...
[5570] chr11 [32995613, 32995652] +
[5571] chr11 [32995685, 32995987] +
[5572] chr11 [32996154, 32996292] +
[5573] chr11 [32996611, 32996640] +
[5574] chr11 [32996794, 32997176] +
---
seqlengths:
chr11
NA

> bedDanRer7Fn <- file.path(system.file("extdata", package="CNEr"),
+                             "filter_regions.danRer7.bed")
> bedDanRer7 <- readBed(bedDanRer7Fn)

Reading /private/var/folders/lc/_5ks0mjd3gvf_dq14g3thsr40000gn/T/Rtmp83Fap1/Rinst7c9f15bed9b
> bedDanRer7

GRanges with 5000 ranges and 0 metadata columns:
      seqnames          ranges strand
      <Rle>             <IRanges> <Rle>
[1] chr1 [ 38, 194] +
[2] chr1 [1101, 1123] +
[3] chr1 [1254, 1302] +
[4] chr1 [1473, 1508] +
[5] chr1 [1782, 2105] +
...
[4996] chr1 [1720306, 1720403] +
[4997] chr1 [1720405, 1720433] +
[4998] chr1 [1720436, 1720784] +
[4999] chr1 [1720785, 1720988] +
[5000] chr1 [1721374, 1721394] +
---
seqlengths:
chr1
NA

```

3.3 Chromosome size of query assembly

The chromosome size of the query assembly is necessary when the filters for query assembly is provided.

To facilitate the fetch of chromosome sizes, we prepared a function `fetchChromSizes` to automate the procedure. Currently the assemblies from UCSC are

supported, and more assemblies from other sources will be implemented in the future. A object of `Seqinfo` is returned which suits the input to downstream analysis.

```
> qSizesHg19 <- fetchChromSizes("hg19")
> qSizesDanRer7 <- fetchChromSizes("danRer7")

> qSizesHg19
```

Seqinfo of length 93

seqnames	seqlengths	isCircular	genome
chr1	249250621	<NA>	hg19
chr2	243199373	<NA>	hg19
chr3	198022430	<NA>	hg19
chr4	191154276	<NA>	hg19
chr5	180915260	<NA>	hg19
...
chrUn_gl000231	27386	<NA>	hg19
chrUn_gl000229	19913	<NA>	hg19
chrM	16571	<NA>	hg19
chrUn_gl000226	15008	<NA>	hg19
chr18_gl000207_random	4262	<NA>	hg19

```
> qSizesDanRer7
```

Seqinfo of length 1133

seqnames	seqlengths	isCircular	genome
chr7	77276063	<NA>	danRer7
chr5	75682077	<NA>	danRer7
chr3	63268876	<NA>	danRer7
chr4	62094675	<NA>	danRer7
chr1	60348388	<NA>	danRer7
...
Zv9_NA959	1142	<NA>	danRer7
Zv9_NA937	1137	<NA>	danRer7
Zv9_NA802	1126	<NA>	danRer7
Zv9_NA1000	942	<NA>	danRer7
Zv9_NA997	650	<NA>	danRer7

(Seqinfo) can also be generated from local two bit file with `seqinfo` from *rtracklayer*.

4 CNE identification

In this section, we will go through the details of CNE identification.

4.1 net alignments scan

Detecting CNEs highly relies on the whole-genome pairwise net alignments. To correct the bias of a chosen genome and capture the duplicated CNEs during genome evolution, we scan two sets of nets for each pairwise comparison, one as reference from each of the genomes.

We identify CNEs by scanning the alignments for regions with at least **I** identities over **C** alignment columns. Because different genes and loci may favor various similarity scores, we usually scan at two different window sizes 30 and 50 with several similarity criterias (**I/C**) range from 70% to 100%.

```
> ## axt, GRanges objects as input
> 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"))
> ## axt and bed files as input
> CNEHg19DanRer7 <- ceScan(axts=axtFilesHg19DanRer7, tFilter=bedHg19Fn,
+                           qFilter=bedDanRer7Fn, qSizes=qSizesDanRer7,
+                           thresholds=c("45_50", "48_50", "49_50"))
> CNEDanRer7Hg19 <- ceScan(axts=axtFilesDanRer7Hg19, tFilter=bedDanRer7Fn,
+                           qFilter=bedHg19Fn, qSizes=qSizesHg19,
+                           thresholds=c("45_50", "48_50", "49_50"))
```

At this stage, a (list) of (data.frame) is returned from (ceScan), which contains the preliminary CNEs. Here is some exemple output:

```
> #data(CNEHg19DanRer7)
> lapply(CNEHg19DanRer7, head)

$`45_50`
  tName  tStart    tEnd qName  qStart    qEnd strand score
1 chr11 31085937 31085991 chr7 16911672 16911726      - 92.73
2 chr11 31182676 31182730 chr25 15074418 15074472      + 89.09
3 chr11 31182806 31182932 chr25 15074676 15074804      + 89.15
4 chr11 31257974 31258025 chr25 15098901 15098951      + 90.38
5 chr11 31263401 31263511 chr25 15100713 15100823      + 92.79
6 chr11 31623054 31623103 chr25 15188234 15188283      + 90.00

      cigar
1          55M
2          55M
3 81M1D35M1D11M
4       32M1I19M
5          111M
6          50M
```



```
$`48_50`
      tName   tStart     tEnd qName   qStart     qEnd strand score cigar
1 chr11 31263427 31263497 chr25 15100739 15100809      + 95.77   71M
2 chr11 31685844 31685902 chr25 15243641 15243699      + 96.61   59M
3 chr11 31712869 31712922 chr25 15247368 15247421      + 96.30   54M
4 chr11 31734350 31734412 chr25 15252105 15252167      + 96.83   63M
5 chr11 31785866 31785928 chr25 15255878 15255940      + 96.83   63M
6 chr11 31785931 31785985 chr25 15255943 15255997      + 96.36   55M
```

```
$`49_50`
      tName   tStart     tEnd qName   qStart     qEnd strand score cigar
1 chr11 31785866 31785928 chr25 15255878 15255940      + 96.83   63M
2 chr11 32053036 32053091 chr25 15333377 15333432      + 98.21   56M
```

In the result table, even though the strand for query element can be negative, the coordiante for that query element is already on the positive strand.

4.2 Merge CNEs

Because we do two rounds of CNE detections with each genome as reference, some conserved elements overlap on both genomes and are supposed to be removed. But elements that only overlap on one of the genomes are kept, so that duplicated elements remain distinct.

```
> #data(CNEDanRer7Hg19)
> cneMergedDanRer7Hg19 <- mapply(cneMerge, CNEDanRer7Hg19, CNEHg19DanRer7,
+                               SIMPLIFY=FALSE)
> lapply(cneMergedDanRer7Hg19, head)
```

```
$`45_50`
      chr1   start1     end1 chr2   start2     end2 strand similarity
19 chr25 15381678 15381819 chr11 32221854 32221994      +      89.44
20 chr25 15397394 15397480 chr11 31821057 31821143      +      89.66
21 chr25 15403140 15403217 chr11 31829491 31829568      +      92.31
22 chr25 15415208 15415296 chr11 31848222 31848310      +      89.89
23 chr25 15418715 15418765 chr11 31989683 31989732      +      90.20
24 chr25 15432800 15433014 chr11 32053010 32053222      +      91.63
      cigar
19    114M1I27M
20         87M
21         78M
22         89M
23    22M1I28M
24 82M1I32M1I99M
```

```
$`48_50`
```

	chr1	start1	end1	chr2	start2	end2	strand	similarity
11	chr25	15381700	15381752	chr11	32221876	32221928	+	96.23
12	chr25	15403140	15403193	chr11	31829491	31829544	+	96.30
13	chr25	15432826	15432883	chr11	32053036	32053092	+	96.55
14	chr25	15432887	15432970	chr11	32053096	32053178	+	96.43
15	chr25	15459924	15459973	chr11	32198103	32198152	+	96.00
17	chr25	15100739	15100809	chr11	31263427	31263497	+	95.77

cigar

11	53M
12	54M
13	56M1I1M
14	28M1I55M
15	50M
17	71M

\$`49_50`

	chr1	start1	end1	chr2	start2	end2	strand	similarity
3	chr25	15432826	15432881	chr11	32053036	32053091	+	98.21
4	chr25	15255878	15255940	chr11	31785866	31785928	+	96.83
5	chr25	15333377	15333432	chr11	32053036	32053091	+	98.21

cigar

3	56M
4	63M
5	56M

4.3 Realignment of CNEs

Some CNEs might be unannotated repeats. To remove them, currently we use **blat** (Kent, 2002) to realign each sequence of CNEs against the respective genomes. When the number of matches exceed certain threshold, for instance, 8, that CNE will be discarded.

This step can be very time-consuming when the number of CNEs are large. Other alignment method can also be considered, such as Bowtie2, BWA. The two bit file for each assembly is required.

```
> 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=8L, cutoffs2=8L,
+             assembly1Twobit=assemblyDanRer7Twobit,
+             assembly2Twobit=assemblyHg19Twobit,
+             blatBinary="blat")
+ }
```

```
+ }
```

Now at this stage, these elements are the final CNEs. We also prepare a one step function `ceScanOneStep` and it returns a CNE object directly which wraps all the necessary information. This one-step function is highly recommended to avoid the tedious steps above.

```
> assemblyHg19Twobit <- "/Users/gtan/CSC/CNEr/2bit/hg19.2bit"
> assemblyDanRer7Twobit <- "/Users/gtan/CSC/CNEr/2bit/danRer7.2bit"
> finalCNE <- ceScanOneStep(axt1=axtHg19DanRer7, filter1=bedHg19,
+                           sizes1=qSizesHg19, assembly1="hg19",
+                           twoBit1=assemblyHg19Twobit,
+                           axt2=axtDanRer7Hg19, filter2=bedDanRer7,
+                           sizes2=qSizesDanRer7, assembly2="danRer7",
+                           twoBit2=assemblyDanRer7Twobit,
+                           thresholds=c("45_50", "48_50", "49_50"),
+                           blatBinary="blat", blatCutoff1=8L, blatCutoff2=8L)
```

5 CNE storage and query

As the computation of CNEs from the whole pipeline and the preparation of annotation package can be very time-consuming, for a smoother visualisation experience, we decided to use a local SQLite database to store these information.

5.1 CNE storage and query

Since the CNEs `data.frame` is just a table and can be imported into a SQL table naturally. To speed up the query from the SQL database, the bin indexing system is acquired. For more information, please refer to the paper ([Kent et al., 2002](#)) and [genomewiki](#).

```
> ## on individual tables
> 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)
+ }
> ## on CNE class
> data(finalCNE)
> saveCNEToSQLite(finalCNE, dbName=dbName, overwrite=TRUE)
```

When querying results from the local SQLite database based on the chr, coordinates and other criterias, a `IRanges` object is returned.

```

> chr <- "chr11"
> start <- 31000000L
> end <- 33000000L
> minLength <- 50L
> tableName <- "danRer7_hg19_45_50"
> fetchedCNERanges <- readCNERangesFromSQLite(dbName, tableName, chr,
+                                             start, end, whichAssembly="L",
+                                             minLength=minLength)

```

6 CNEs visualisation

To visualise the CNEs together with other gene annotations, we choose to use the Bioconductor package *Gviz* in this vignette. *Gviz*, based on the *grid* graphics scheme, is a very powerful package for plotting data and annotation information along genomic coordinates. The functionality of integrating publicly available genome annotation data, such as UCSC or Ensembl, significantly reduced the burden of preparing annotations for common assemblies. Since the Bioconductor release 2.13 of *Gviz*, it provides the data track in horizon plot, which exactly meets our needs for visualisation of CNEs density plots. For more detailed usage, please check the vignette or manual of *Gviz*.

Another option for visualisation is the package *ggbio*, which is based on *ggplot2*. The advantage of *ggbio* is the simplicity of adding any customised *ggplot2* style track into the plot without tuning the coordinate systems. The densities generated in the following section can be easily plot in the horizon plot. A short straightforward tutorial regarding horizon plot in *ggplot2* format is available from <http://timelyportfolio.blogspot.co.uk/2012/08/horizon-on-ggplot2.html>.

6.1 Gene annotation visualisation

For the example of hg19 vs danRer7 in this vignette, we choose hg19 as the reference and show the range of developmental gene **PAX6**.

```

> library(Gviz)
> genome <- "hg19"
> chr <- "chr11"
> start <- 31000000L
> end <- 33000000L
> axisTrack <- GenomeAxisTrack()
> ideoTrack <- IdeogramTrack(genome=genome, chromosome=chr)
> cpgIslands <- UcscTrack(genome=genome, chromosome=chr,
+                         track="cpgIslandExt", from=start, to=end,
+                         trackType="AnnotationTrack", start="chromStart",
+                         end="chromEnd", id="name", shape="box",
+                         fill="#006400", name="CpG Islands")

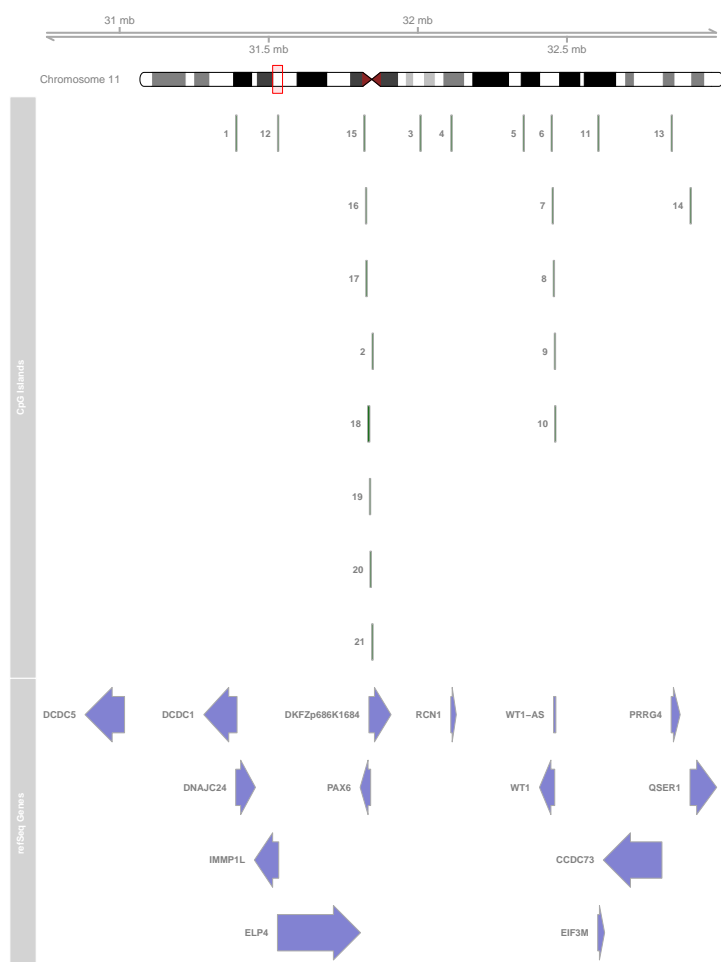
```

```

> refGenes <- UcsTrack(genome="hg19", chromosome=chr,
+                      track="refGene", from=start, to=end,
+                      trackType="GeneRegionTrack", rstarts="exonStarts",
+                      rends="exonEnds", gene="name2", symbol="name2",
+                      transcript="name", strand="strand", fill="#8282d2",
+                      name="refSeq Genes", collapseTranscripts=TRUE)
> biomTrack <- BiomartGeneRegionTrack(genome="hg19", chromosome=chr,
+                                     start=start, end=end, name="Ensembl")

> library(Gviz)
> plotTracks(list(axisTrack, ideoTrack, cpgIslands, refGenes),
+            collapseTranscripts=TRUE, shape="arrow",
+            showId=TRUE, transcriptAnnotation="symbol")

```



It is also possible to plot the annotation from an ordinary RR object, such as `asdata.frame`, `GRanges`, `IRanges` or even from a local file. Usually the `gff` file

containing the gene annotation can be processed by *Gviz* directly. For more details, please look into the vignette of *Gviz*.

6.2 CNEs horizon plot

```
> dbName <- file.path(system.file("extdata", package="CNEr"),
+                        "cne.sqlite")
> 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_48_50",
+               assembly1="hg19", chr=chr, start=start,
+               end=end, windowSize=windowSize,
+               minLength=minLength)
> cneHg19DanRer7_49_50 <-
+   CNEDensity(dbName=dbName,
+               tableName="danRer7_hg19_49_50",
+               assembly1="hg19", chr=chr, start=start,
+               end=end, windowSize=windowSize,
+               minLength=minLength)

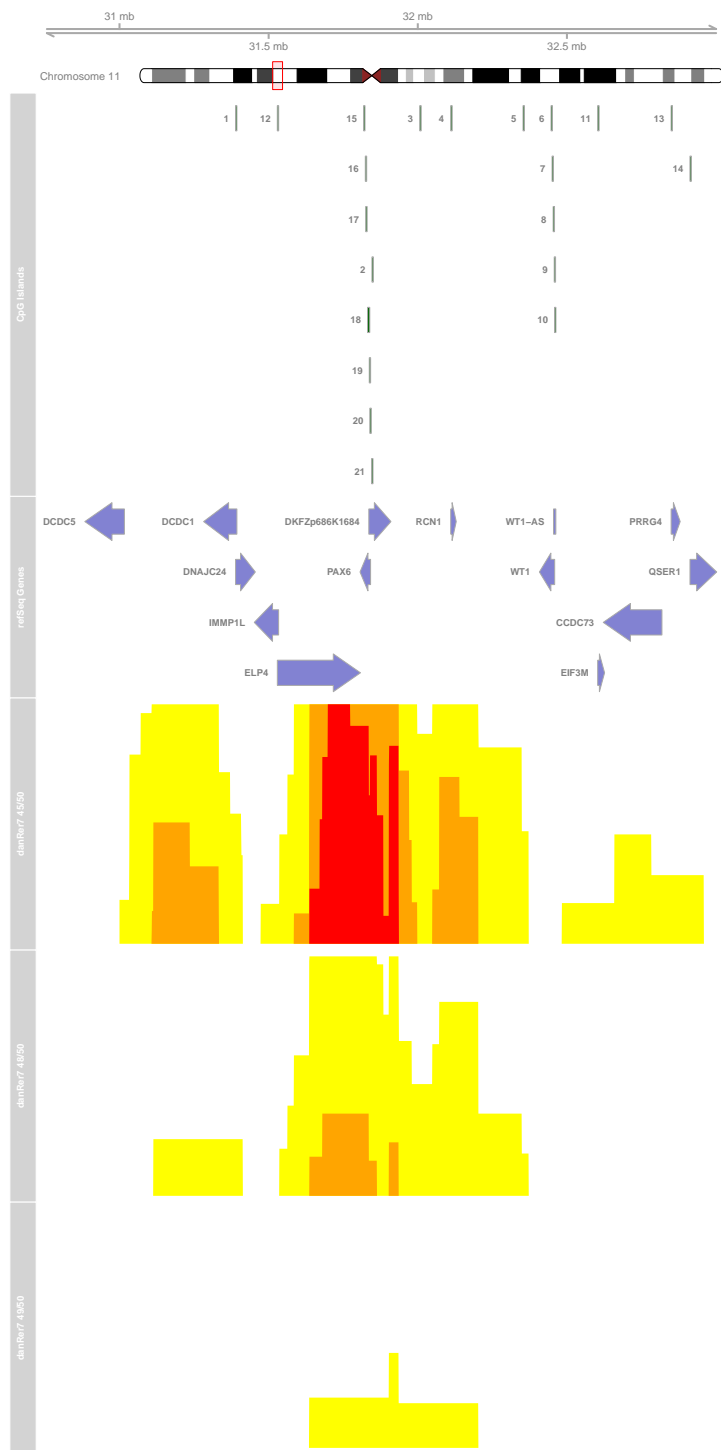
> data(cneHg19DanRer7_45_50)
> data(cneHg19DanRer7_48_50)
> data(cneHg19DanRer7_49_50)
> genome <- "hg19"
> chr <- "chr11"
> start <- 31000000L
> end <- 33000000L
> #axisTrack = GenomeAxisTrack()
> #ideoTrack = IdeogramTrack(genome=genome, chromosome=chr)
> strand <- "+"
> dataMatrix <- cneHg19DanRer7_45_50
> dTrack1 <- DataTrack(start=dataMatrix[,1], end=dataMatrix[,1],
+                       data=dataMatrix[,2], chromosome=chr, strand=strand,
+                       genome=genome, type="horiz", horizon.scale=0.1,
+                       fill.horizon=c("#B41414", "#E03231", "#F7A99C",
+                                       "yellow", "orange", "red"),
+                       name="danRer7 45/50")
> dataMatrix <- cneHg19DanRer7_48_50
```

```

> dTrack2 <- DataTrack(start=dataMatrix[,1], end=dataMatrix[,1],
+                       data=dataMatrix[,2], chromosome=chr, strand=strand,
+                       genome=genome, type="horiz", horizon.scale=0.1,
+                       fill.horizon=c("#B41414", "#E03231", "#F7A99C",
+                                     "yellow", "orange", "red"),
+                       name="danRer7 48/50")
> dataMatrix <- cneHg19DanRer7_49_50
> dTrack3 <- DataTrack(start=dataMatrix[,1], end=dataMatrix[,1],
+                       data=dataMatrix[,2], chromosome=chr, strand=strand,
+                       genome=genome, type="horiz", horizon.scale=0.1,
+                       fill.horizon=c("#B41414", "#E03231", "#F7A99C",
+                                     "yellow", "orange", "red"),
+                       name="danRer7 49/50")

> plotTracks(list(axisTrack, ideoTrack, cpgIslands,
+                 refGenes,
+                 dTrack1, dTrack2, dTrack3),
+             collapseTranscripts=TRUE, shape="arrow",
+             showId=TRUE, transcriptAnnotation="symbol")

```



From this horizon plot compared with Zebrafish with Human as reference genome, the developmental gene PAX6 was surrounded by the density peaks of CNEs.

7 Conclusion

With this package, we are able to identify CNEs efficiently and handle the corresponding objects conveniently in R. Horizon plot shows a superior dynamic range to the standard density plots, simultaneously revealing CNE clusters characterized by vastly different levels of sequence conservation. Such CNE density plots generated using precise locations of CNEs can be used to identify genes involved in developmental regulation, even for novel genes that are not yet annotated.

The following is the session info that generated this vignette:

```
> sessionInfo()

R version 3.1.0 (2014-04-10)
Platform: x86_64-apple-darwin13.1.0 (64-bit)

locale:
[1] C/en_US.UTF-8/en_US.UTF-8/C/en_US.UTF-8/en_US.UTF-8

attached base packages:
[1] parallel  grid      stats      graphics  grDevices  utils
[7] datasets  methods  base

other attached packages:
[1] XVector_0.4.0      IRanges_1.22.5      Gviz_1.8.0
[4] BiocGenerics_0.10.0 CNEr_1.0.0

loaded via a namespace (and not attached):
[1] AnnotationDbi_1.26.0    BBmisc_1.6
[3] BSgenome_1.32.0        BatchJobs_1.2
[5] Biobase_2.24.0         BiocParallel_0.6.0
[7] Biostrings_2.32.0      DBI_0.2-7
[9] Formula_1.1-1          GenomeInfoDb_1.0.2
[11] GenomicAlignments_1.0.1 GenomicFeatures_1.16.0
[13] GenomicRanges_1.16.3   Hmisc_3.14-4
[15] R.methodsS3_1.6.1      RColorBrewer_1.0-5
[17] RCurl_1.95-4.1         RSQLite_0.11.4
[19] Rcpp_0.11.1            Rsamtools_1.16.0
[21] VariantAnnotation_1.10.0 XML_3.98-1.1
[23] biomaRt_2.20.0         biovizBase_1.12.1
[25] bitops_1.0-6           brew_1.0-6
[27] cluster_1.15.2         codetools_0.2-8
```

[29]	colorspace_1.2-4	dichromat_2.0-0
[31]	digest_0.6.4	fail_1.2
[33]	foreach_1.4.2	iterators_1.0.7
[35]	lattice_0.20-29	latticeExtra_0.6-26
[37]	matrixStats_0.8.14	munsell_0.4.2
[39]	plyr_1.8.1	rtracklayer_1.24.0
[41]	scales_0.2.4	sendmailR_1.1-2
[43]	splines_3.1.0	stats4_3.1.0
[45]	stringr_0.6.2	survival_2.37-7
[47]	tools_3.1.0	zlibbioc_1.10.0

References

- Kent WJ (2002). “BLAT—the BLAST-like alignment tool.” *Genome Research*, **12**(4), 656–664.
- Kent WJ, Sugnet CW, Furey TS, Roskin KM, Pringle TH, Zahler AM, Haussler aD (2002). “The Human Genome Browser at UCSC.” *Genome Research*, **12**(6), 996–1006.
- Sandelin A, Bailey P, Bruce S, Engström PG, Klos JM, Wasserman WW, Ericson J, Lenhard B (2004). “Arrays of ultraconserved non-coding regions span the loci of key developmental genes in vertebrate genomes.” *BMC genomics*, **5**(1), 99.
- Woolfe A, Goodson M, Goode DK, Snell P, McEwen GK, Vavouri T, Smith SF, North P, Callaway H, Kelly K (2004). “Highly conserved non-coding sequences are associated with vertebrate development.” *PLoS biology*, **3**(1), e7.