rnaSeqMap: RNASeq analyses using xmap database back-end

Anna Lesniewska, Michal Okoniewski

May 15, 2011

Vignette for v.2.7.11 - created without XMAP database access.

Contents

1 Recent changes and updates 2

2 Introduction 2

3 Database preparation 3

4 Analysis 4

5 ReadSet and NucleotideDistr classes 4

6 Connecting to data from BAM files 6

7 Region Mining algorithm 6

8 Interesting gene criteria 7

8.1 Coverage 7

8.2 Splicing 8

8.3 Fold change in regions 9

8.4 Region-based coverage 9

8.5 Utilities for transforming NucleotideDistr objects 11

9 Analyzing whole chromosomes 11

10 Producing output for DESeq and edgeR 13

11 Visualization 13

1 Recent changes and updates

14.05.2011 - added parseGff3() and bam2sig() - MO, AL

2 Introduction

rnaSeqMap is a "middleware" library for RNAseq secondary analyses. It constitutes an API for such operations as:

- access to any the reads of the experiment in possibly fastest time, according to any chromosome coordinates
- accessing sets of reads according to genomic annotation in Ensembl
- calculation of coverage and number of reads and transformations of those values
- creating input for significance analysis algorithms - from edgeR and DESeq
- precisely finding significant and consistent regions of expression
- splicing analyses
- visualizations of genes and expression regions

The library is independent from the sequencing technology and reads mapping software. It needs either reads described as genome coordinates in the extended xmap database, or can alternatively read data as big as they can fit in the operational memory. The use with modified xmap database is recommended, as it overcomes memory limitations - thus the library can be efficiently run on not very powerful machines.

The internal features of rnaSeqMap distinctive for this piece of software are:

- sequencing reads and annotations in one common database - extended XMAP [2]
- algorithm for finding irreducible regions of genomic expression - according to Aumann and Lindell [1]
- nucleotide-level splicing analysis
- connectors for further gene- and region-level expression processing to DESeq [5] and edgeR [6]
- the routines for coverage, splicing index and region mining algorithm have been implemented in C for speed

The area of application of the library may be adjusted to the type of experiment performed. Mainly, it is intended to analyze the data from RNAseq with depth-of-coverage estimation of expression, while it can be also used for SAGE, Chip-Seq, copy number analysis, etc. The functions in an appropriate context may be used not only for gene-level and region-level expression estimation, but also eg. for quality control of sequencing data.

The library follows the paradigms of the exonmap package (see [4]), continued currently as xmapcore. The point is: with the use of the annotation database, it gives a framework to analyze genes and intergenic regions gene by gene, filter the genes according to their qualities and visualize the internals of region expression. Due to the qualities of sequencing data, `rnaSeqMap` may be used not only to genes, but also to all the regions on the genome. The approach does not do any summarization in the initial phases so it is possible to have the fine-grained access by nucleotide (eg. as in `NucleotideDistr` class).

3 Database preparation

The database which is used in the analysis is xmapcore database, extended by additional relational tables (defined with indexing and partitioning) and stored procedures. First, an instance of xmapcore database has to be installed, then the script for adding the tables `seq_reads` and `seq_experiments` have to be run (see `rnaSeqMap.sql` script in the `inst scripts`) The table seq_reads is filled with the reads from the sequencers, which may be for example data from .SAM files transformed by sam2xmap.awk script

```
awk -f sam2xmap.awk input.sam > seq_reads.txt
```

The content of the awk script is as follows :

```
# AWK script to convert the SAM file to seq_read table input for X:MAP
# For mouse 19 chromosomes + 20=X,21=Y,22=MT
{
  OFS = "\t"
  strand = rshift($2,4)  % 2
  if (strand==0) strand = -1
  cc = substr($3,4)
  if (cc!=cc+0)
   {
    if (cc=="X") cc=20
    else if (cc=="Y") cc=21
    else if (cc=="M") cc=22
    else cc=30;
   }

  print(6, cc,  $4, $4+ length($10),strand)
  #remember to change first param to the number of sample!!
}
```

As the chromosome numbers are simplified in the database into numbers, the X, Y and mitochondrial chromosomes are coded as consecutive number after the last numbered chromosome - as in the example for mouse above. In the `rnaSeqMap` in R, they are recoded according to the species information set by selecting xmapcore database or by `setSpecies()`

After such preparation, all the data files must be loaded to the database, using `mysqlimport`, eg:

```
myslqimport -u mysqlexampleuser -p xmapcore_homo_sapiens_58 -local seq_reads
```

The `seq_reads` table has then to be indexed as in the script `inst`
`scripts`
`indices.sql`.

The table bio_sample has to be created. It contains experiment description: all the samples in seq_reads are identified by an integer number, while in bio_sample the numbers have the treatments and other experimental attributes assigned. The content of this table is assigned to the phenoData slot of the NucleotideDistr objects that are processed.

4 Analysis

After loading the rnaSeqMap library, connecting to xmap database is done with xmap.connect.

```
> library(rnaSeqMap)
```

```
> xmap.connect("human-24")
```

Sample objects, used for analysis described in this vignette, may be found in the dataset:

```
> data(sample_data_rnaSeqMap)
> ls()

[1] "rs.list"

> class(rs.list)

[1] "list"

> length(rs.list)

[1] 17

> names(rs.list)

 [1] "ENSG00000164287" "ENSG00000118515" "ENSG00000146859" "ENSG00000174469"
 [5] "ENSG00000002726" "ENSG00000146540" "ENSG00000164877" "ENSG00000214783"
 [9] "ENSG00000165071" "ENSG00000185920" "ENSG00000133816" "ENSG00000129151"
[13] "ENSG00000137494" "ENSG00000220703" "ENSG00000249935" "ENSG00000177757"
[17] "ENSG00000223972"
```

The list contains 17 example genes with their coverage from a real experiment that contains 6 samples. Alternatively, the objects can be fed with data directly from BAM files, as described in the next sections.

5 ReadSet and NucleotideDistr classes

ReadSet is an initial container for sequencing reads. Using the database filled with reads, the SeqReads objects are created by specifying their genome coordinates and filling them with reads information:

```
> rs <- newSeqReads(g.ch, st, en, str)
> rs <- addExperimentsToReadset(rs, 1:6)
```

There are functions that create SeqReads objects for genes:

```
> rs <- newSeqReadsFromGene(g)
> rs <- addExperimentsToReadset(rs, 1:6)
```

The table bio_sample has to be created. It contains experiment description: all the samples in seq_reads are identified by an integer number, while in bio_sample the numbers have the treatments and other experimental attributes assigned. The content of this table is assigned to the phenoData slot of the NucleotideDistr objects that are processed.

```
> setSpecies("homo_sapiens")
```

```
> rs <- rs.list[[1]]
> rs <- newSeqReads("chr2", 220238268, 220264744, -1)
> rs <- newSeqReads(rnaSeqMap:::.chr.convert(2), 220143989, 220151622,
+     1)
> f <- c("test1.bam", "test2.bam", "test3.bam", "test4.bam", "test5.bam")
> ff <- sapply(f, function(x) system.file("extdata", x, package = "rnaSeqMap"))
> rs <- getBamData(rs, 1:5, files = ff)
> nd.cov <- getCoverageFromRS(rs, 1:5)
> dd <- RleList2matrix(distribs(nd.cov))
> dd[990:1000, ]
```

```
      [,1] [,2] [,3] [,4] [,5]
 [1,]    3    2    0    2    0
 [2,]    3    2    0    2    0
 [3,]    3    2    0    2    0
 [4,]    3    2    0    2    0
 [5,]    3    2    0    2    0
 [6,]    3    2    0    2    0
 [7,]    3    2    0    2    0
 [8,]    3    2    0    2    0
 [9,]    3    2    0    2    0
[10,]    3    2    0    2    0
[11,]    3    2    0    2    0
```

The distributions in an object may be normalized by getting the sum of reads for each sample from the database or by any other scaling vector:

```
> nsums <- getSumsExp()
```

```
> nsums
```

```
[1]  25918  32577  63250 123445 126977
```

```
> nd.cov <- normalizeBySum(nd.cov, nsums)
> RleList2matrix(distribs(nd.cov))[990:1000, ]
```

```
      [,1] [,2] [,3] [,4] [,5]
 [1,]    3    2    0    0    0
 [2,]    3    2    0    0    0
 [3,]    3    2    0    0    0
 [4,]    3    2    0    0    0
 [5,]    3    2    0    0    0
 [6,]    3    2    0    0    0
 [7,]    3    2    0    0    0
 [8,]    3    2    0    0    0
 [9,]    3    2    0    0    0
[10,]    3    2    0    0    0
[11,]    3    2    0    0    0
```

6 Connecting to data from BAM files

The way of using the ReadSet and NucleotideDistr with data directly from BAM files is using the covdesc file, which is used as the experiment description. Covdesc is assigned as a phenoData table of the objects.

Here, the sample BAM files including data from chromosome 2 and examplary genes: TUBA4A,PTPRN,MIR153-1,DNPEP,CHPF,PAX3, FARSB will be used.

7 Region Mining algorithm

The region mining algorithm is an adaptation of the sliding-and-combing two window algorithm by Lindell and Aumann [1]. Originally, it was used to find quantitative association rules from datasets with two numerical attributes. In the case of genomic data, especially RNAseq, the X dimension is the genome coordinate in bp, the Y dimension (decisive attribute) is the level of coverage for a given nucleotide. By specifying the μ and *minsup* parameters, we may tune it to find regions of a given magnitude of coverage and minimal length.

The regions found by this algorithm are proven to be irreducible. This means here, that if the region has mean coverage over μ, then when divided into any two sub-regions, those sub-regions will both have also mean coverage higher than μ. This follows the intuition of a region with a biologically consistent expression - namely - an exon. So if one applies properly tuned Lindell algorithm to the RNAseq data, should be able to find precise boundaries of real transcription.

From a NucleotideDistr object, another object may be created that includes the distributions equal 0 when the part of the distribution is not within irreducible region, or appropriate μ-level when the nucleotide in a given sample has a coverage that makes it a part of an irreducible region:

```
> nd.reg <- findRegionsAsND(nd.cov, 6, 10)
> RleList2matrix(distribs(nd.reg))[990:1000, ]
```

```
      [,1] [,2] [,3] [,4] [,5]
 [1,]    0    0    0    0    0
 [2,]    0    0    0    0    0
 [3,]    0    0    0    0    0
 [4,]    0    0    0    0    0
 [5,]    0    0    0    0    0
 [6,]    0    0    0    0    0
 [7,]    0    0    0    0    0
 [8,]    0    0    0    0    0
 [9,]    0    0    0    0    0
[10,]    0    0    0    0    0
[11,]    0    0    0    0    0
```

Alternatively, regions from a single distribution may be also presented as an `RangedData` object:

```
> findRegionsAsIR(nd.cov, 6, 10, 1)

IRanges of length 5
        start       end width
[1] 220144081 220144124    44
[2] 220144547 220144644    98
[3] 220144792 220144845    54
[4] 220145018 220145068    51
[5] 220145307 220151561  6255
```

8 Interesting gene criteria

The genes may be regarded as interesting of significant according to many features, that can be calculated using their `NucleotideDistr` values.

8.1 Coverage

Coverage is included in the object (access with `distr()`), however in the heuristics for finding genes, we various statistical measures of the coverage for each sample:

```
> dd <- RleList2matrix(distribs(nd.cov))
> apply(dd, 2, max)

[1] 169  16  17   3   2

> apply(dd, 2, mean)

[1] 11.4736704  1.7549122  1.4996070  0.2226880  0.1683259

> apply(dd, 2, sum)

[1] 87590 13397 11448  1700  1285

> apply(dd, 2, sd)

[1] 19.1742518  2.5280929  2.7922639  0.5092400  0.4392479
```

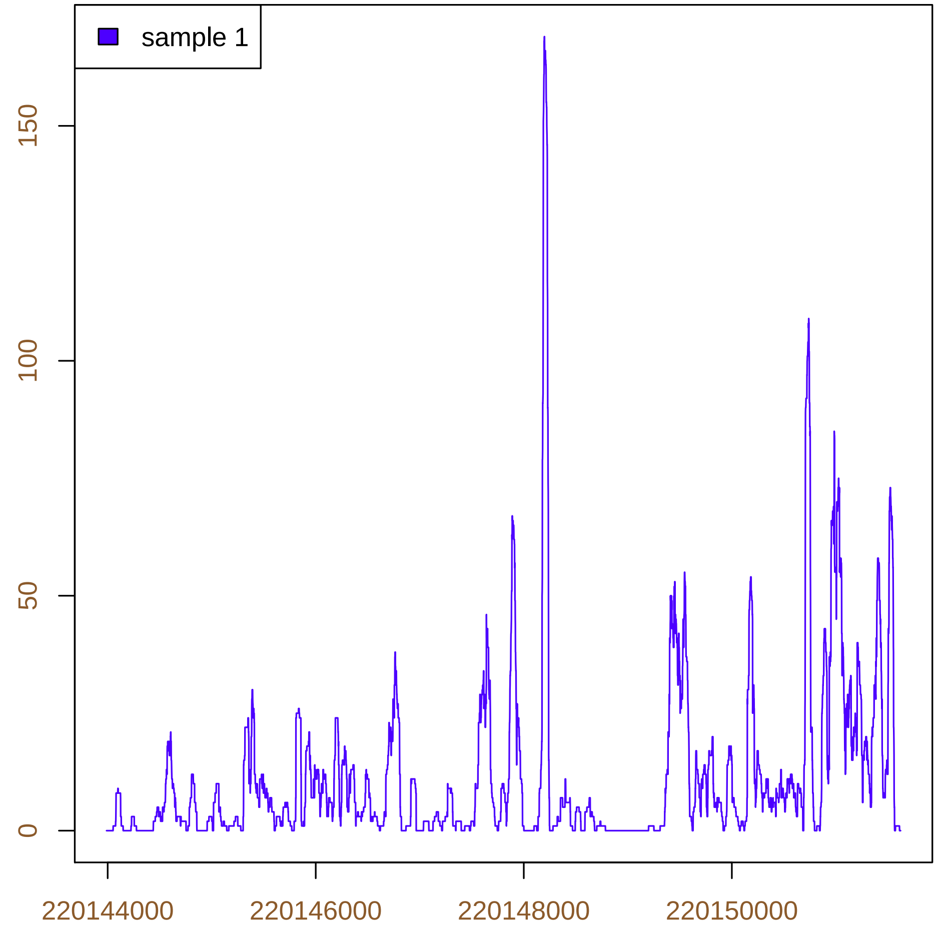
8.2 Splicing

We use a modified version of splicing index from [3] paper. The value of index is calculated for each nucleotide in the investigated region. Gene level proportion is calculated using sum of coverage functions in the region for both treatments.

However, many nucleotides have no coverage in one or both conditions. Often the coverage in one condition is much higher than another. Thus the value of splicing index is set to 1 or -1 when second or first conditions have zero coverage. In the case when the log2 value exceeds the range (-1,1) it is limited to -1 or 1 respectively.

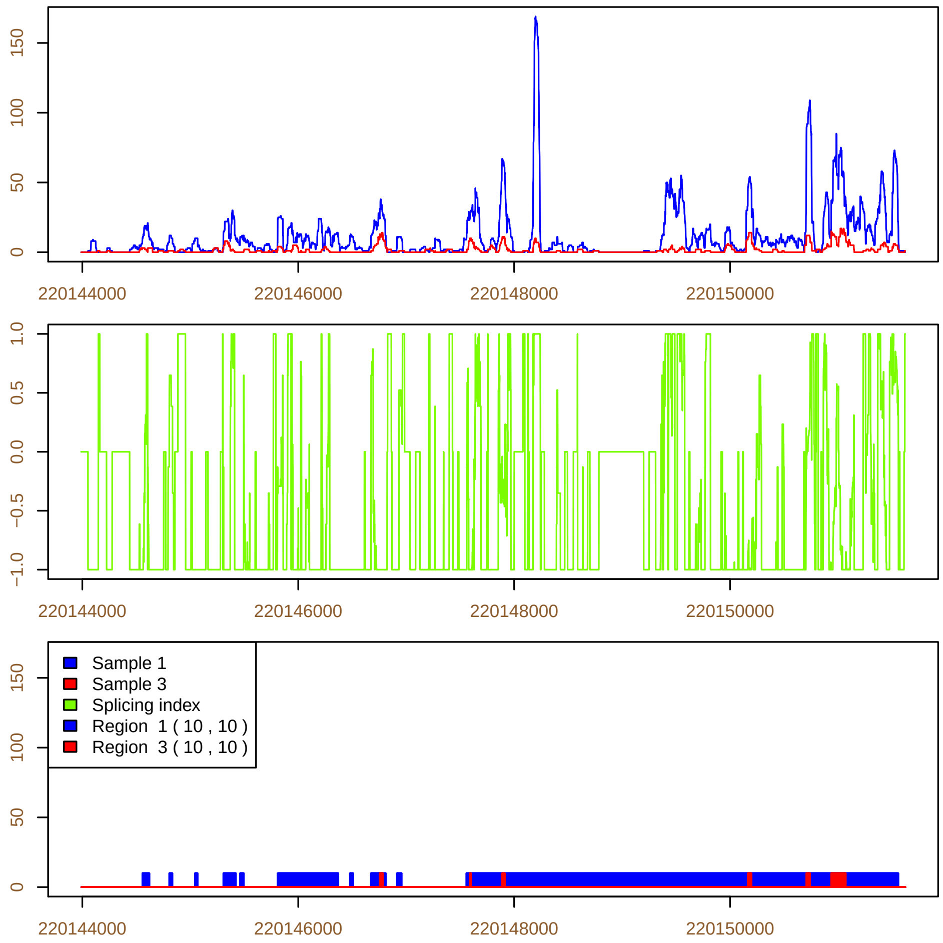
```
> nd.si <- getSIFromND(nd.cov, c(2, 3))
> RleList2matrix(distribs(nd.si))[990:1000, ]
```

```
> distrCOVPlot(nd.cov, 1)
```



Splicing index may be plotted for both genes and any genomic regions:

```
> distrSIPlot(nd.cov, 1, 3, 10, 10)
```



```
> plotSI(2, 223436000, 223437000, -1, 2, 4)
```

8.3 Fold change in regions

For the genes that have enough coverage, it is possible to calculate nucleotide-level fold change using:

```
> nd.fc <- getFCFromND(nd.cov, c(2, 3))
> RleList2matrix(distribs(nd.fc))[990:1000, ]

 [1] 2 2 2 2 2 2 2 2 2 2 2
```

8.4 Region-based coverage

Region-based coverage is a function of discretizing the `NucleotideDistr` object according to a sequence of Lindell algorithm searches.It assigns to the resulting `NucleotideDistr` the value of maximal μ from a sequence for which there is a region covering it

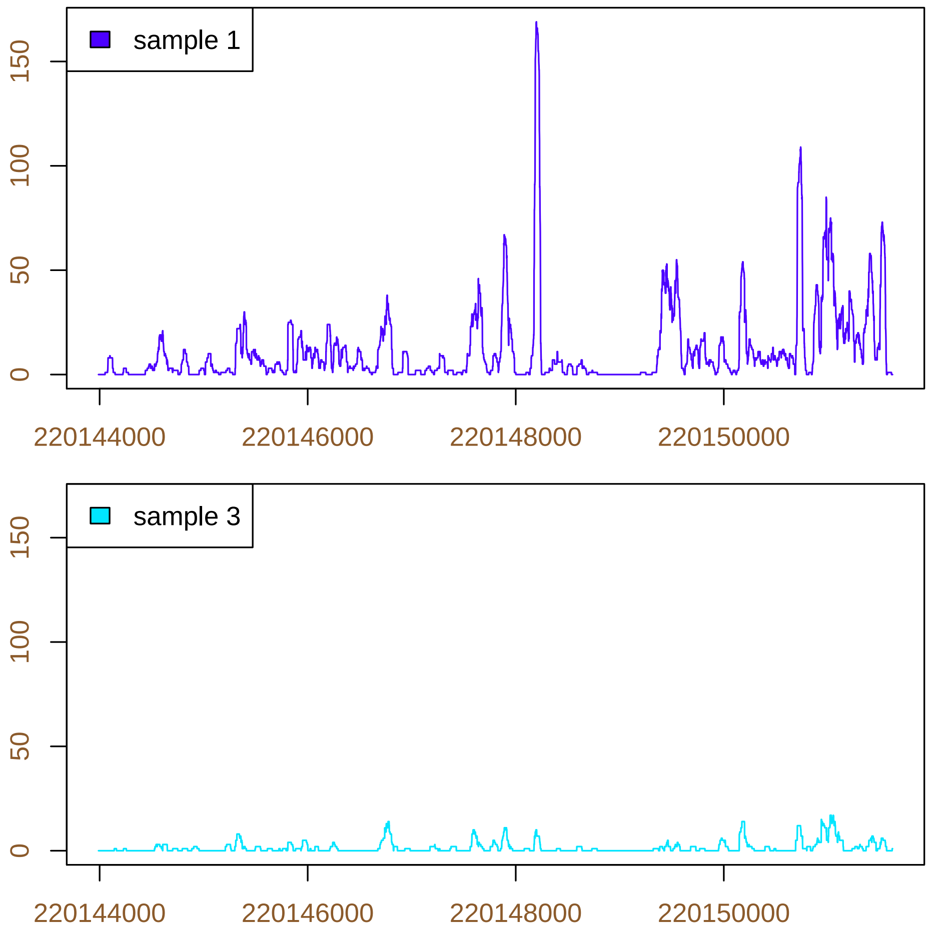
```
> nd.rbcov <- regionBasedCoverage(nd.cov, seqq = 1:10, maxexp = 40,
+      minsup = 10)
> RleList2matrix(distribs(nd.rbcov))[990:1000, ]
```

```
      [,1] [,2] [,3] [,4] [,5]
 [1,]    4    0    0    0    0
 [2,]    4    0    0    0    0
 [3,]    4    0    0    0    0
 [4,]    4    0    0    0    0
 [5,]    4    0    0    0    0
 [6,]    4    0    0    0    0
 [7,]    4    0    0    0    0
 [8,]    4    0    0    0    0
 [9,]    4    0    0    0    0
[10,]    4    0    0    0    0
[11,]    4    0    0    0    0
```

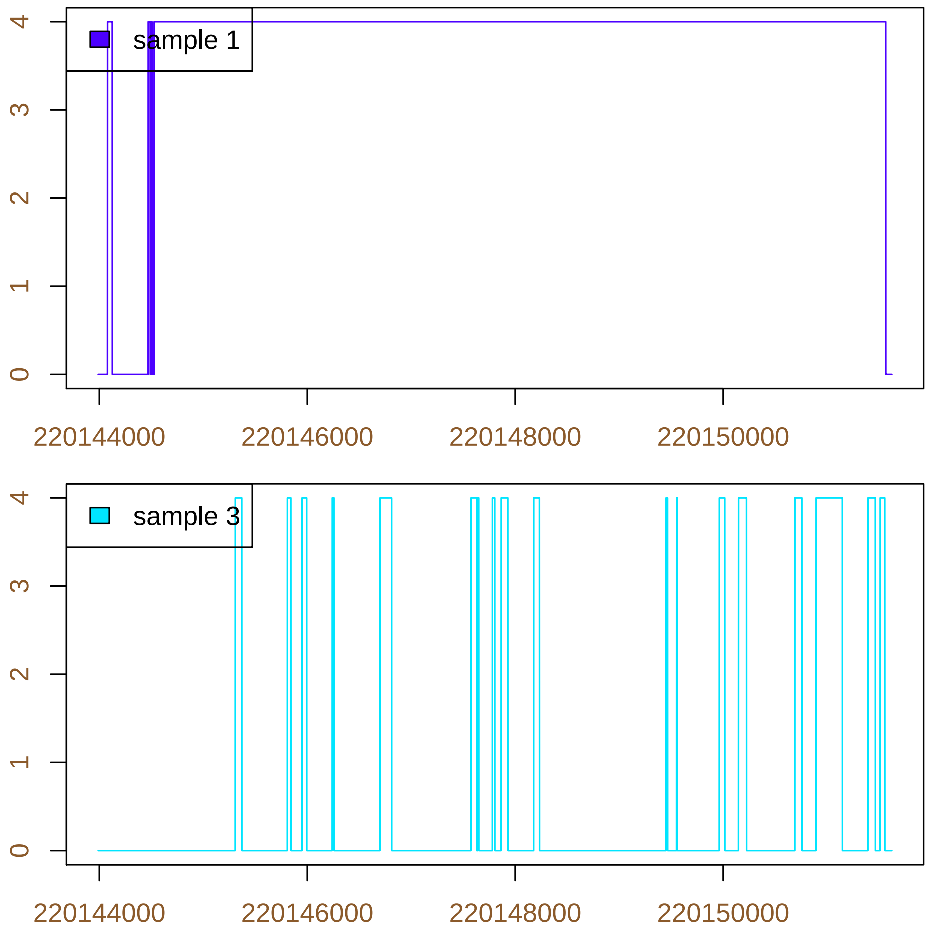
Such a distribution is useful for removing artifacts from the distributions: high peaks and sudden drops of signal.

The plots are depicting the coverage before and after the region-based transformation.

```
> distrCOVPlot(nd.cov, c(1, 3))
```



```
> distrCOVPlot(nd.rbcov, c(1, 3))
```



8.5 Utilities for transforming `NucleotideDistr` objects

Additional utilities to operate on `NucleotideDistr` objects are functions for combining two, averaging, getting the sum or log2 for all the elements.

9 Analyzing whole chromosomes

The functions`spaceInChromosome` and `geneInChromosome` are finding all the genes and intergenic spaces in given regions. Such a table of genes or spaces may be used to build a genome-wide pipeline for finding interesting gene expression or novel areas of intergenic expression (possibly novel genes).

```
> my.genes <- geneInChromosome(22, 2e+07, 20400000, 1)
> my.genes
> my.spaces <- spaceInChromosome(22, 2e+07, 20400000, 1)
> my.spaces
```

The example of function used to evaluate genes or spaces:

```
> test.gene <- function(g, exps, nsums, mi = 5, minsup = 15) {
+     rs <- newSeqReadsFromGene(g)
+     rs <- addExperimentsToReadset(rs, exps)
+     cat("added reads..\n")
+     nd.cov <- getCoverageFromRS(rs, exps)
+     cat("calculated coverage...\n")
+     nd.cov <- normalizeBySum(nd.cov, nsums)
+     cat("normalized...\n")
+     nd.reg <- findRegionsAsND(nd.cov, as.integer(mi), minsup = minsup)
+     ir.reg <- findRegionsAsIR(nd.cov, as.integer(mi), minsup = minsup)
+     cat("ran region search algorithm...\n")
+     print(ir.reg)
+     out <- g
+     out <- c(out, apply(distribs(nd.cov), 2, max))
+     out <- c(out, apply(distribs(nd.cov), 2, mean))
+     out <- c(out, apply(distribs(nd.reg), 2, max))
+     out
+ }
> test.space <- function(exps, ch, st, en, str, nsums, mi = 5,
+     minsup = 15) {
+     g.ch <- rnaSeqMap:::.chromosome.number(ch)
+     rs <- newSeqReads(g.ch, st, en, str)
+     rs <- addExperimentsToReadset(rs, exps)
+     nd.cov <- getCoverageFromRS(rs, exps)
+     nd.cov <- normalizeBySum(nd.cov, nsums)
+     cat("Running ch ", g.ch, " start", st, "\n")
+     nd.reg <- findRegionsAsND(nd.cov, as.integer(mi), minsup = minsup)
+     out <- c(ch, st, en, str)
+     out <- c(out, apply(distribs(nd.cov), 2, max))
+     out <- c(out, apply(distribs(nd.cov), 2, mean))
+     out <- c(out, apply(distribs(nd.reg), 2, max))
+     out
+ }
> interesting.genes <- NULL
> for (i in 1:length(my.genes)) {
+     cat("Running gene ", i, "---------------------\n")
+     interesting.genes <- rbind(interesting.genes, test.gene(my.genes[i],
+         1:6, nsums))
+ }
> interesting.spaces <- NULL
> for (i in 104:(dim(my.spaces))[1]) {
+     cat("Running space ", i, "---------------------\n")
+     interesting.spaces <- rbind(interesting.spaces, test.space(1:2,
+         22, my.spaces[i, 1], my.spaces[i, 2], my.spaces[i, 3]))
+ }
```

10 Producing output for DESeq and edgeR

The functions buildDESeq and buildDGEList are the connectors between rnaSeqMap and the analysis of expression on the gene level as implemented in DESeq [5] and edgeR [6].

11 Visualization

rnaSeqMap has a set of chromosome-based plots that may be used for distributions contained in a NucleotideDistr object or just a region defined by chromosome coordinates - taking data from the database and not creating any objects.

Examples of the plots:

Histogram-based distribution of coverage:

```
> plotCoverageHistogram(2, 223435255, 223521056, -1, 1, 500)
```

Coverage plots for a gene, with official exons of the gene plotted below:

```
> distrCOVPlotg(names(rs.list)[16], 1:3)
```

References

[1] Aumann Y and Lindell Y, A Statistical Theory for Quantitative Association Rules, Journal of Intelligent Information Systems, 2003

[2] Yates T, Okoniewski M and Miller C, X:Map: annotation and visualization of genome structure for Affymetrix exon array analysis, Nucleic Acids Research, 2007

[3] Gardina P et al, Alternative splicing and differential gene expression in colon cancer detected by a whole genome exon array, BMC Genomics, 2006

[4] Okoniewski M et al, An annotation infrastructure for the analysis and interpretation of Affymetrix exon array data, Genome Biology, 2007

[5] Anders S and Huber W, Differential expression analysis for sequence count data, Nature Preceedings, 2010

[6] Robinson M et al, edgeR: a Bioconductor package for differential expression analysis of digital gene expression data, Bioinformatics, Bioinformatics, 2009