# rnaSeqMap

October 25, 2011

---

`NDplots` *Genomic plots based upon NucleotideDistr objects*

---

### Description

Various plots of genomic coverage for data from `NucleotideDistr` objects

### Usage

```
distrCOVPlot(nd,exps)
distrSIPlot(nd, ex1, ex2, mi,minsup=5)
```

### Arguments

| | |
|---|---|
| `nd` | `NucleotideDistr` object |
| `exps` | vectors of experiment numbers to plot |
| `ex1,ex2` | experiment numbers to plot |
| `mi` | threshold in the region mining algorithm |
| `minsup` | minimal support - minimal length of the irreducible region found |

### Author(s)

Michal Okoniewski, Anna Lesniewska

### Examples

```
data(sample_data_rnaSeqMap)
rs <- rs.list[[1]]
 if (xmapConnected())
 {
  nd.cov <- getCoverageFromRS(rs,1:6)
  distrSIPlot(nd.cov,  1,3, mi=5, minsup=10)
 }
```

NucleotideDistr-class *Numeric distributions by nucleotide - class*

## Description

An S4 class that inherits from `eSet` and holds all the numeric distributions of functions defined over the genome. The values may include coverage, splicing, fold change, etc. for a region defined by genomic coordinates.

## Slots/List Components

Objects of this class contain (at least) the following list components:

`chr`: numeric matrix containing the read counts.

`start`: data.frame containing the library size and group labels.

`end`: data.frame containing the library size and group labels.

`strand`: data.frame containing the library size and group labels.

`start`: data.frame containing the library size and group labels.

## Methods

`distribs` gives the matrix of distributions from `assayData`

`getDistr` gives a single distributions from `assayData` as a vector

`newNuctleotideDistr (distribs, chr, start, end, strand, type="UNKNOWN", phenoData=NULL, featureData=NULL)` constructor from a matrix of data and chromosome coordinates.

## Author(s)

Anna Lesniewska, Michal Okoniewski

## See Also

SeqReads, NDtransforms

RleList2matrix *RleList2matrix*

## Description

Function transforms list of Rle objects to matrix.

## Usage

```
RleList2matrix(list);
```

### Arguments

| | |
|---|---|
| list | list of Rle objects. |

### Value

Produces the full, unpacked coverage matrix from a list of Rle objects. Used to re-format the coverage data.

### Author(s)

Michal Okoniewski, Anna Lesniewska

### Examples

```
if (xmapConnected())
{
  rs <- newSeqReads(1,1,20000,1)
  rs <- getBamData(rs,1:3)
  nd.cov <- getCoverageFromRS(rs,1:3)
  RleList2matrix(nd.cov@data)
}
```

---

## SeqReads *SeqReads - a container for RNAseq reads*

---

### Description

SeqReads objects keep the reads information in the form of a list, containing one matrix of reads per experiment. Matrices of dimension n x 2 should come from a mapping to the regions defined by genome coordinates (chromosome, start, end, strand) in the SeqReads object.

The object may be filled in from the database or from list with read data. It is recommended to create one SeqReads object per gene or intergenic region. The object are used then ot create object of class NucleotideDistr

### Usage

```
newSeqReads(chr, start, end, strand, datain=NULL, phenoData=NULL, featureData=NU
newSeqReadsFromGene(g)
```

### Arguments

| | |
|---|---|
| chr | Chromosome |
| start | Start of the region on a chromosome |
| end | End of the region on a chromosome |
| strand | Genome strand: 1 or -1 |
| datain | If supplied, it must be a list of matrices of reads start and stop |
| g | Ensembl identifier of a gene |
| phenoData | |
| featureData | |
| covdesc | Filename for experiment description |

## Value

Object of a class SeqReads

## Author(s)

Michal Okoniewski, Anna Lesniewska

---

`addBamData` *addBamData - getting sample data from BAM file.*

---

## Description

Add data from experimental samples stored in BAM file.

## Usage

```
addBamData(rs, file, exp, phenoDesc=NULL)
```

## Arguments

| | |
|---|---|
| `rs` | SeqReads object to modify |
| `file` | BAM file to read |
| `exp` | Numbers of sample slot in the object |
| `phenoDesc` | A vector to add to phenoData |

## Value

SeqReads object with samples added from the BAM files. List of BAM files comes from the covdesc. The covdesc content becomes phenoData of the object.

## Author(s)

Michal Okoniewski, Anna Lesniewska

## Examples

```
if (xmapConnected())
{
  rs <- newSeqReads(1,1,20000,1)
  rs <- addBamData(rs,1:3)
}
```

`addDataToReadset` *addDataToReadset - adding one more sample in the SeqRead on R level*

## Description

Add another reads matrix to the readset. No control of region consistency, the matrix needs just 2 columns: starts and ends.

## Usage

```
addDataToReadset(rs, datain, spl)
```

## Arguments

| | |
|---|---|
| `rs` | |
| `datain` | |
| `spl` | Number or name of the experimental sample |

## Value

SeqReads object with one more sample added.

## Author(s)

Michal Okoniewski, Anna Lesniewska

## Examples

```
rs <- newSeqReads(1,1,20000,1)
my.data1 <- rbind(c(1,50), c(3,53), c(11,60))
rs <- addDataToReadset(rs, my.data1, 1)
```

---

`addExperimentsToReadset` *addExperimentsToReadset - getting sample data from the database.*

## Description

Add data from experimental samples in the xXMAP database to the readset. Connection to the database required.

## Usage

```
addExperimentsToReadset(rs, exps)
```

## Arguments

| | |
|---|---|
| `rs` | SeqReads object to modify |
| `exps` | Vector of numbers of experimental samples in xXMAP |

## Value

SeqReads object with samples added from the database.

## Author(s)

Michal Okoniewski, Anna Lesniewska

## Examples

```
if (xmapConnected())
{
  rs <- newSeqReads(1,1,20000,1)
  rs <- addExperimentsToReadset(rs,1:3)
}
```

---

`averageND` *averageND, sumND, combineNS, log2ND - operations on distributions*

---

## Description

Set of functions to operate on `NucleotideDistr` objects.

averageND calculates the mean for samples, sumND adds up selected samples' distributions, combineND adds two objects with the same size of distribution matrix, log2ND transforms all numeric data in the object into log space.

## Usage

```
averageND(nd, exps);
sumND(nd, exps);
combineND(nd1, nd2);
log2ND(nd);
```

## Arguments

| | |
|---|---|
| `nd, nd1, nd2` | `NucleotideDistr` objects |
| `exps` | a pair of numbers of samples in the experiment |

## Value

`NucleotideDistr` object of the same type as input objects

## Author(s)

Michal Okoniewski, Anna Lesniewska

### Examples

```
if (xmapConnected())
{
  rs <- newSeqReads(1,1,20000,1)
  nd.cov <- getCoverageFromRS(rs,1:3)
  nd.avg <- averageND(nd.cov,c(1,3))
  nd.sum <- averageND(nd.cov,c(1,3))
  nd.sum <- combineND(nd.cov,nd.cov)
  nd.log <- log2ND(nd.cov)
}
```

---

`bam2sig` *bam2sig - encapsulated pipeline of finding significant expression*

---

### Description

Reads BAM files according to annotation and produces output table processed with DESeq negative binomial test.

### Usage

```
bam2sig(annot, covdesc="covdesc", species="homo_sapiens")
```

### Arguments

| | |
|---|---|
| `annot` | Character table or data frame with colums: chr, start, end, strand, name |
| `covdesc` | Name of the file that includes BAM files (experiment description file) |
| `species` | Species name - needed for .chr.convert function - to match BAM and annotation chromosome names |

### Value

Output table with significant expression results, as from DESeq

### Author(s)

Michal Okoniewski, Anna Lesniewska

### Examples

```
if (xmapConnected())
{
 all.g <- all.genes(as.vector=F)
 ss <- sample(1:20000, 10)
 genes <- as.data.frame(all.g[ss,])

 genes <- cbind(as.vector(genes[,"stable_id"]), as.vector(genes[,"space"]), as.vector(g
colnames(genes) <- c("name", "chr", "start", "end", "strand")

 deseqRes <- bam2sig()
 deseqRes[1:10,]
}
```

`buildDESeq` *buildDESeq - create CountDataSet*

## Description

Creates `CountDataSet` from the data in the database using the list of genes supplied - for further analysis with DESeq

## Usage

```
buildDESeq(genes,exps,conds=NULL)
```

## Arguments

| | |
|---|---|
| `genes` | vector of Ensembl gene IDs |
| `exps` | vector of experiments |
| `conds` | Vector of experimental condition descriptions for the samples |

## Value

`CountDataSet` object filled with the data of gene-level counts of reads

## Author(s)

Michal Okoniewski, Anna Lesniewska

## See Also

buildDGEList

## Examples

```
if (xmapConnected())
{
  data(sample_data_rnaSeqMap)
  gg <- names(rs.list)
  cds <- buildDESeq(gg,1:6, c("a","b","b","a","a","b"))
}
```

`buildDGEList` *buildDGEList - create DGEList (edgeR)*

## Description

Creates `DGEList` from the data in the database using the list of genes supplied - for further analysis with edgeR

## Usage

```
buildDGEList(genes,exps,conds=NULL)
```

### Arguments

| | |
|---|---|
| genes | vector of Ensembl gene IDs |
| exps | vector of experiments |
| conds | Vector of experimental condition descriptions for the samples |

### Value

DGEList object filled with the data of gene-level counts of reads

### Author(s)

Michal Okoniewski, Anna Lesniewska

### See Also

buildDESeq

### Examples

```
if (xmapConnected())
{
  data(sample_data_rnaSeqMap)
  gg <- names(rs.list)
  cds <- buildDGEList(gg,1:6, c("a","b","b","a","a","b"))
}
```

---

## findRegionsAsIR *findRegionsAsIR - finding regions of high coverage using Lindell-Aumann*

---

### Description

The function is running Lindell-Aumann algorithm to find regions of irreducible expression on the coverage data in the NucleotideDistr object. The function may be used to find the location and boundaries of significant expression of exons and small RNA.

### Usage

```
findRegionsAsIR(nd, mi, minsup=5, exp)
```

### Arguments

| | |
|---|---|
| nd | An object of NucleotideDistr class that has coverage values for a given region |
| mi | The threshold of coverage that makes the region significant |
| minsup | Minimal support of the numeric association rule - namely, in this case, the mininmal length of the discovered region |
| exp | Sample (experiment) number |

## Value

IRanges object with irreducible regions with high coverage.

## Author(s)

Michal Okoniewski, Anna Lesniewska

## Examples

```
if (xmapConnected())
{
  rs <- newSeqReads(1,1,20000,1)
  rs <- addExperimentsToReadset(rs,1:3)
  nd.cov <- getCoverageFromRS(rs,1:3)
  nd.regs <- findRegionsAsND(nd.cov, 10)
}
```

---

`findRegionsAsND` *findRegionsAsND - finding regions of high coverage using Lindell-Aumann*

---

## Description

The function is running Lindell-Aumann algorithm to find regions of irreducible expression on the coverage data in the `NucleotideDistr` object. The function may be used to find the location and boundaries of significant expression of exons and small RNA.

## Usage

```
findRegionsAsND(nd, mi, minsup=5)
```

## Arguments

| | |
|---|---|
| `nd` | An object of `NucleotideDistr` class that has coverage values for a given region |
| `mi` | The threshold of coverage that makes the region significant |
| `minsup` | Minimal support of the numeric association rule - namely, in this case, the mininmal length of the discovered region |

## Value

NucleotideDistr object that includes a matrix with zeros where no region was found and the value of mi for all the nucleotides included in the region. The type fo the object is "REG".

## Author(s)

Michal Okoniewski, Anna Lesniewska

### Examples

```
if (xmapConnected())
{
  rs <- newSeqReads(1,1,20000,1)
  rs <- addExperimentsToReadset(rs,1:3)
  nd.cov <- getCoverageFromRS(rs,1:3)
  nd.regs <- findRegionsAsND(nd.cov, 10)
}
```

---

## geneInChromosome *geneInChromosome*

---

### Description

Finds all the genes in the given chromosome regions

### Usage

```
geneInChromosome(chr, start, end, strand)
```

### Arguments

| | |
|---|---|
| chr | Chromosome |
| start | Start of the region on a chromosome |
| end | End of the region on a chromosome |
| strand | Genome strand: 1 or -1 |

### Value

table of the genes in a given regions, produced with stored procedure

### Author(s)

Michal Okoniewski, Anna Lesniewska

### Examples

```
if (xmapConnected())
{
  geneInChromosome(1, 1, 80000, 1)
}
```

`getBamData` *getBamData - getting sample data from BAM file.*

## Description

Add data from experimental samples stored in BAM file.

## Usage

```
getBamData(rs, exps=NULL, files=NULL, unstranded=FALSE, covdesc="covdesc")
```

## Arguments

| | |
|---|---|
| `rs` | SeqReads object to modify |
| `exps` | Vector of numbers of experimental samples |
| `files` | Vector of BAM files to read |
| `unstranded` | Flag which type of data are using (with distinguishing strand or not) |
| `covdesc` | Alternatively, the experiment description file |

## Value

SeqReads object with samples added from the BAM files. List of BAM files comes from the covdesc. The covdesc content becomes phenoData of the object.

## Author(s)

Michal Okoniewski, Anna Lesniewska

## Examples

```
if (xmapConnected())
{
  rs <- newSeqReads(1,1,20000,1)
  rs <- getBamData(rs,1:3)
}
```

`getCoverageFromRS` *getCoverageFromRS - conversion to coverage object*

## Description

Calculates the coverage function for the reads encapsulated in the SeqReads object.

## Usage

```
getCoverageFromRS(rs, exps)
```

### Arguments

| | |
|---|---|
| rs | SeqReads object to modify |
| exps | Vector of numbers of experimental samples in xXMAP |

### Value

NucleotideDistr object with coverage matrix in assayData slot and type "COV".

### Author(s)

Michal Okoniewski, Anna Lesniewska

### Examples

```
if (xmapConnected())
{
  rs <- newSeqReads(1,1,20000,1)
  rs <- addExperimentsToReadset(rs,1:6)
  nd.cov <- getCoverageFromRS(rs,1:3)
}
```

---

## getExpDescription *getExpDescription*

---

### Description

Gets the bio_sample table from the database. May be used as phenoData.

### Usage

```
getExpDescription()
```

### Value

Table of experimental factors assigned to numbers of samples.

### Author(s)

Michal Okoniewski, Anna Lesniewska

getFCFromND *getFCFromND - calculating fold change of coverages*

## Description

This function calculates the fold change of two sample coverages from a NucleotideDistr objects. The coverages are assumed to be after logarithmic transformation, so the function basically subtracts the value and generates new NucleotideDistr object with a single vector of fold changes.

## Usage

```
getFCFromND(nd, exps)
```

## Arguments

| | |
|---|---|
| nd | NucleotideDistr object with coverages |
| exps | a pair of numbers of samples in the experiment |

## Value

NucleotideDistr object of type "FC" with a single vector of fold changes

## Author(s)

Michal Okoniewski, Anna Lesniewska

## Examples

```
if (xmapConnected())
{
  rs <- newSeqReads(1,1,20000,1)
  rs <- addExperimentsToReadset(rs,1:3)
  nd.cov <- getCoverageFromRS(rs,1:3)
  nd.fc <- getFCFromND(nd.cov,c(1,3))
}
```

getSIFromND *getSIFromND - calculating splicing index of two coverages*

## Description

This function calculates the splicing index value of two sample coverages from a NucleotideDistr object. It is assumed that the region in the NucleotideDistr is a single gene. Splicing index is calculated in similar way to the implementation for exon Affy microarrays (see Gardina et al, Genome Biology, 2007 for details), but it is run for each nucleotide in the region and instead of gene-level average expression values, it uses sums of reads for both samples.

## Usage

```
getSIFromND(nd, exps)
```

### Arguments

| | |
|---|---|
| nd | NucleotideDistr object with coverages |
| exps | a pair of numbers of samples in the experiment |

### Value

NucleotideDistr object of type "FC" with a single vector of splicing index values

### Author(s)

Michal Okoniewski, Anna Lesniewska

### Examples

```
if (xmapConnected())
{
  rs <- newSeqReads(1,1,20000,1)
  nd.cov <- getCoverageFromRS(rs,1:3)
  nd.fc <- getSIFromND(nd.cov,c(1,3))
}
```

---

## getSumsExp *getSumsExp*

---

### Description

Gets the sum of reads in all the samples present in the database in the seq_read table

### Usage

```
getSumsExp()
```

### Value

Vector of sums

### Author(s)

Michal Okoniewski, Anna Lesniewska

### Examples

```
if (xmapConnected())
{
 sums <- getSumsExp()
 nsums
}
```

normalizeBySum *Normalization of NucleotideDistr by global number of reads*

## Description

normalizeBySum function normalizes the coverage values in NucleotideDistr by dividing all the numbers for all samples by the sum of reads for each sample. The number of reads from each sample may be taken from the database by the function getSumsExp, which is a wrapper for an appropriate SQL procedure. Alternatively, it is passed directly as a vector of numeric values of the same length as the number of samples analyzed. Such simple normalization allows comparisons of the coverage values for samples with different number of reads

## Usage

```
normalizeBySum(nd,r=NULL)
```

## Arguments

| | |
|---|---|
| nd | NucleotideDistr object with raw read counts |
| r | Vector of numbers. If there is no such parameter, a database procedure summarizing reads is run |

## Value

NucleotideDistr object

## Author(s)

Michal Okoniewski, Anna Lesniewska

## See Also

getSumsExp

## Examples

```
if (xmapConnected())
{
  rs <- newSeqReads(1,10000,20000,1)
  nd.cov <- getCoverageFromRS(rs,1:3)
  nd.norm <- normalizeBySum(nd.cov)
  nd.norm <- normalizeBySum(nd.cov, r=c(100, 200, 1000))
}
```

`parseGff3` *parseGff3 - parsing gff3 file format*

## Description

Parses gff3 file into genes, transcripts and exons.

## Usage

```
parseGff3(filegff, fileg="genes.txt", filet="transcripts.txt", filee="exons.txt"
```

## Arguments

| | |
|---|---|
| `filegff` | Input file in GFF3 format |
| `fileg` | Filename for output: genes |
| `filet` | Filename for output: transcripts |
| `filee` | Filename for output: exons |
| `nofiles` | Flag: just optput list, no files |

## Value

List with elements "genes", "transcripts", "exons" with appropriate tables.

## Author(s)

Michal Okoniewski, Anna Lesniewska

## Examples

```
if (xmapConnected())
{
   parseGff3("Athaliana.gff3")
}
```

`plotGeneCoverage` *Genomic plots with rnaSeqMap*

## Description

Various plots of genomic coverage for experiments.

## Usage

```
plotGeneCoverage(gene_id, ex)
plotRegionCoverage(chr, start, end, strand, ex)
plotExonCoverage (exon_id,ex)
plotCoverageHistogram (chr,start,end,strand,ex, skip)
plotGeneExonCoverage(gene_id, ex)
plotSI(chr,start,end,strand, exp1, exp2 )
```

### Arguments

| | |
|---|---|
| ex | vectors of experiment numbers to plot |
| exp1, exp2 | experiment numbers for splicing index |
| gene_id | Ensembl gene ID |
| exon_id | Ensembl exon ID |
| chr | Chromosome |
| start | Start position of region on the chromosome |
| end | Start position of region on the chromosome |
| strand | Strand |
| skip | size of the bucket in histogram |

### Author(s)

Michal Okoniewski, Anna Lesniewska

### Examples

```
if (xmapConnected())
{
  plotGeneCoverage( "ENSG00000144567", 1:3) # plotting FAM134A for experiments 1,2,3
  plotRegionCoverage( 2, 220040947, 220050201, 1, 1:3 ) # the same, using coordinates
}
```

---

## readsInRange *readsInRange*

---

### Description

Finds all the reads for a genomic range

### Usage

```
readsInRange(chr, start, end, strand, ex)
```

### Arguments

| | |
|---|---|
| chr | Chromosome |
| start | Start of the region on a chromosome |
| end | End of the region on a chromosome |
| strand | Genome strand: 1 or -1 |
| ex | experiment |

### Value

table of reads, as in the database

### Author(s)

Michal Okoniewski, Anna Lesniewska

## Examples

```
if (xmapConnected())
{
 tmp <- readsInRange( 1, 10000, 20000, 1,3)
}
```

---

`regionBasedCoverage`

*regionBasedCoverage - transformation of the region coverage by the*

---

## Description

The function builds a `NucleotideDistr` object from another object of coverage, using sequential call of Lindell-Aumann algorithm on the same data with a sequence of mi-levels. Each nucleotide is assigned the maximum mi-value of a region that covers it.

The output `NucleotideDistr` object has the distribution without peaks and small drops of coverage, but the thade-off is that the level of coverage are discrete: seq*maxexp.

## Usage

```
regionBasedCoverage(nd, seqq=1:10, maxexp=20, minsup=5)
```

## Arguments

| | |
|---|---|
| `nd` | An object of `NucleotideDistr` class that has coverage values for a given region |
| `seqq` | Vector of numbers used to divide the range of coverage for subsequent mi-levels |
| `maxexp` | The maximal mi-level for coverage |
| `minsup` | Minimal support of the numeric association rule - namely, in this case, the mininmal length of the discovered region |

## Value

NucleotideDistr object that includes a matrix with zeros where no region was found and a maximum of mi-levels used for the sequential region searched. The distributions are similar to coverage, but have removed outliers of coverage peaks and short drops of coverage.

## Author(s)

Michal Okoniewski, Anna Lesniewska

## Examples

```
if (xmapConnected())
{
rs <- newSeqReads(1,1,20000,1)
rs <- addExperimentsToReadset(rs,1:3)
nd.cov <- getCoverageFromRS(rs,1:3)
nd.regs <- regionBasedCoverage(nd.cov, 1:10, 100)
#runs the Lindell-Aumann algorithm at 100, 90, ... and picks maximal mi-level, where th
}
```

`regionCoverage` *regionCoverage*

## Description

Finds all the reads for a genomic range

## Usage

```
regionCoverage(chr, start, end, strand, ex, db = "FALSE" )
```

## Arguments

| | |
|---|---|
| `chr` | Chromosome |
| `start` | Start of the region on a chromosome |
| `end` | End of the region on a chromosome |
| `strand` | Genome strand: 1 or -1 |
| `ex` | experiment |
| `db` | Use database (SQL) implementation of the algorithm |

## Value

coverage vector, independent from `NucleotideDistr`

## Author(s)

Michal Okoniewski, Anna Lesniewska

## Examples

```
if (xmapConnected())
{
 tmp <- regionCoverage( 1, 10000, 20000, 1,3)
}
```

`rs.list` *Example of sequencing data for rnaSeqMap library*

## Description

A fragment of sequencing data from 6 samples - human.

## Usage

```
data(sample_data_rnaSeqMap)
```

## Format

A list with 17 SeqReads objects,each with sequencing reads from 6 samples sequenced with ABI SOLID machine.

### Examples

```
data(sample_data_rnaSeqMap)
length(rs.list)
gene1rs <- rs.list[[1]]
```

---

`setSpecies` *setSpecies*

---

### Description

Sets the species name for chromosomes X, Y and MT translation into consecutive numbers. If you use `xmap.connect`, no need to call `setSpecies`. Both set the internal variable of `xmapcore`.

### Usage

```
setSpecies(name=NULL)
```

### Arguments

| | |
|---|---|
| `name` | Species name |

### Author(s)

Michal Okoniewski, Anna Lesniewska

### Examples

```
setSpecies("mus_musculus")
```

---

`spaceInChromosome` *spaceInChromosome*

---

### Description

Finds all the intergenic spaces in the given chromosome region

### Usage

```
spaceInChromosome(chr, start, end, strand)
```

### Arguments

| | |
|---|---|
| `chr` | Chromosome |
| `start` | Start of the region on a chromosome |
| `end` | End of the region on a chromosome |
| `strand` | Genome strand: 1 or -1 |

### Value

table of the intergenic spaces in a given regions, produced with stored procedure

### Author(s)

Michal Okoniewski, Anna Lesniewska

### Examples

```
if (xmapConnected())
{
  spaceInChromosome(1, 1, 80000, 1)
}
```

---

`xmapConnected` *xmapConnected*

---

### Description

Checks if the connection to the xmap database has been already done. If not, use `xmap.connect`.

### Usage

```
xmapConnected()
```

### Author(s)

Michal Okoniewski, Anna Lesniewska

# Index