# Package 'BRGenomics'

December 28, 2019

**Type** Package

**Title** Tools for the Efficient Analysis of High-Resolution Genomics Data

**Version** 0.2.0

**Description** This package provides useful and efficient utilites for the analysis of high-resolution genomic data using standard Bioconductor methods and classes.

**License** Artistic-2.0

**Encoding** UTF-8

**LazyData** FALSE

**RoxygenNote** 7.0.2

**Depends** rtracklayer,
GenomicFeatures,
BiocParallel

**Imports** GenomicRanges,
stats4,
BiocGenerics,
parallel,
S4Vectors,
IRanges,
AnnotationDbi,
GenomeInfoDb

**Suggests** BiocStyle,
knitr,
rmarkdown,
TxDb.Hsapiens.UCSC.hg38.knownGene,
TxDb.Hsapiens.UCSC.hg19.knownGene,
TxDb.Mmusculus.UCSC.mm10.knownGene,
TxDb.Mmusculus.UCSC.mm9.knownGene,
TxDb.Dmelanogaster.UCSC.dm6.ensGene,
TxDb.Dmelanogaster.UCSC.dm3.ensGene,
testthat

**Enhances** DESeq2

**biocViews** Software,
DataImport,
PROseq,
RNAseq,

ATACseq,
NETseq,
ChIPseq,
CutAndRun,
Transcription,
GeneRegulation,
Normalization

**VignetteBuilder** knitr

# R topics documented:



---

`binNdimensions` *N-dimensional binning of data by quantiles*

---

## Description

This function takes in data along 1 or more dimensions, and for each dimension the data is divided into evenly-sized quantiles from the minimum value to the maximum value, and bin numbers are returned. For instance, if each index of the input data were a gene, the input dimensions would be various quantitative measures of that gene, e.g. expression level, number of exons, length, etc. If plotted in cartesian coordinates, each gene would be a single datapoint, and each measurement would be a separate dimension. The bin numbers for each datapoint in each dimension are returned in a dataframe, with a column for each dimension and a row for each index.

## Usage

```
binNdimensions(..., quantiles = 10)
```

### Arguments

| | |
|---|---|
| ... | A single dataframe, or any number of lists or vectors containing different measurements across the same datapoints. If a dataframe is given, columns should correspond to measurements (dimensions). If lists or vectors are given, they must all have the same lengths. Other input classes will be coerced into a single dataframe. |
| quantiles | Either a number giving the number of quantiles to use for all dimensions (default = 10), or a vector containing the number of quantiles to use for each dimension of input data given. |

### Value

A dataframe containing indices in `1:quantiles` for each datapoint in each dimension.

### Author(s)

Mike DeBerardine

---

## genebodies *Extract Genebodies*

---

### Description

This function returns ranges that are defined relative to the strand-specific start and end sites of regions of interest (usually genes). Unlike `GenomicRanges::promoters`, distances can be upstream or downstream based on the sign, and both the start and end of the returned regions can be defined in terms of either the start of end site of the input ranges. For example, `genebodies(txs,-50,150,fix.end = "start")` is equivalent to `promoters(txs,50,150)`. The default arguments return ranges that begin 300 bases downstream of the original start positions, and end 300 bases upstream of the original end positions.

### Usage

```
genebodies(
  genelist,
  start = 300,
  end = -300,
  fix.start = "start",
  fix.end = "end",
  min.window = 0
)
```

### Arguments

| | |
|---|---|
| genelist | A GRanges object containing genes of interest. |
| start | Depending on `fix.start`, the distance from either the strand-specific start or end site to begin the returned ranges. If positive, the returned range will begin downstream of the reference position; negative numbers are used to return sites upstream of the reference. Set `start = 0` to return the reference position. |

| | |
|---|---|
| end | Identical to the start argument, but defines the strand-specific end position of returned ranges. end must be downstream of start. |
| fix.start | The reference point to use for defining the strand-specific start positions of returned ranges, either "start" or "end". |
| fix.end | The reference point to use for defining the strand-specific end positions of returned ranges, either "start" or "end". Cannot be set to "start" if fix.start = "end". |
| min.window | When fix.start = "start" and fix.end = "end", min.window defines the minimum size (width) of a returned range. However, when fix.end = fix.start, all returned ranges have the same width, and min.window simply size-filters the input ranges. |

## Value

A GRanges object that may be shorter than genelist due to loss of short ranges.

## Author(s)

Mike DeBerardine

## See Also

intra-range-methods

---

# getCountsByPositions *Get signal counts at each position within regions of interest*

---

## Description

Generate a matrix containing a row for each region of interest, and columns for each position (each base if binsize = 1) within each region.

## Usage

```
getCountsByPositions(
  dataset.gr,
  regions.gr,
  binsize = 1,
  FUN = sum,
  simplify.multi.widths = c("list", "pad 0", "pad NA"),
  field = "score",
  ncores = detectCores()
)
```

## Arguments

| | |
|---|---|
| dataset.gr | A GRanges object in which signal is contained in metadata (typically in the "score" field). |
| regions.gr | A GRanges object containing all the regions of interest. |
| binsize | Size of bins (in bp) to use for counting within each range of regions.gr. Note that counts will *not* be length-normalized. |

| | |
|---|---|
| FUN | If binsize > 1, the function used to aggregate the signal within each bin. By default, the signal is summed, but any function operating on a numeric vector can be used. |
| simplify.multi.widths | A string indicating the output format if the ranges in regions.gr have variable widths. Default = "list". See details below. |
| field | The metadata field of dataset.gr to be counted. If length(field) > 1, the output is a list whose elements contain the output for generated each field. |
| ncores | Multiple cores can only be used if length(field) > 1. |

## Details

If the widths of all ranges in regions.gr are equal, a matrix is returned containing a row for each range in regions.gr, and a column for each bin. For input regions.gr with varying widths, setting simplify.multi.widths = "list" will output a list of variable-length vectors, with each vector corresponding to an input region. If simplify.multi.widths = "pad 0" or "pad NA", the output is a matrix containing a row for each range in regions.gr, and a column for each position in each range. The number of columns is determined by the largest range in regions.gr, and columns corresponding to positions outside of each range are either set to 0 or NA, depending on the argument.

## Author(s)

Mike DeBerardine

## See Also

getCountsByRegions

---

# getCountsByRegions *Get signal counts in regions of interest*

---

## Description

Returns a vector the same length as regions.gr containing signal found in each range.

## Usage

```
getCountsByRegions(
  dataset.gr,
  regions.gr,
  field = "score",
  ncores = detectCores()
)
```

## Arguments

| | |
|---|---|
| dataset.gr | A GRanges object in which signal is contained in metadata (typically in the "score" field). |
| regions.gr | A GRanges object containing all the regions of interest. |
| field | The metadata field of dataset.gr to be counted. If length(field) > 1, a dataframe is returned containing the counts for each region in each field. |
| ncores | Multiple cores can only be used if length(field) > 1. |

### Author(s)

Mike DeBerardine

### See Also

getCountsByPositions

---

getDESeqDataSet *Get DESeqDataSet objects for downstream analysis*

---

### Description

This is a convenience function for generating DESeqDataSet objects, but this function also adds support for counting reads across non-contiguous regions.

### Usage

```
getDESeqDataSet(
  dataset.list,
  regions.gr,
  sample_names = names(dataset.list),
  gene_names = NULL,
  sizeFactors = NULL,
  field = "score",
  ncores = detectCores(),
  quiet = FALSE
)
```

### Arguments

| | |
|---|---|
| dataset.list | A list of GRanges datasets that can be individually passed to getCountsByRegions. |
| regions.gr | A GRanges object containing regions of interest. |
| sample_names | Names for each dataset in dataset.list are required, and by default the names of the list elements are used. The names must each contain the string "_rep#", where "#" is a single character (usually a number) indicating the replicate. Sample names across different replicates must be otherwise identical. |
| gene_names | An optional character vector giving gene names, or any other identifier over which reads should be counted. Gene names are required if counting is to be performed over non-contiguous ranges, i.e. if any genes have multiple ranges. If supplied, gene names are added to the resulting DESeqDataSet object. |
| sizeFactors | DESeq2 sizeFactors can be optionally applied in to the DESeqDataSet object in this function, or they can be applied later on, either by the user or in a call to getDESeqResults. Applying the sizeFactors later is useful if multiple sets of factors will be explored, although sizeFactors can be overwritten at any time. |
| field | Argument passed to getCountsByRegions. |
| ncores | Number of cores to use for read counting across all samples. Default is the total number of cores available. |
| quiet | If TRUE, all output messages from call to DESeqDataSet will be suppressed. |

### Value

A DESeqData object in which rowData are given as rowRanges, which are equivalent to regions.gr, unless there are non-contiguous gene regions (see note below). Samples (as seen in colData) are factored so that samples are grouped by replicate and condition, i.e. all non-replicate samples are treated as distinct, and the DESeq2 design = ~condition.

### Use of non-contiguous gene regions

In DESeq2, genes must be defined by single, contiguous chromosomal locations. This function allows individual genes to be encompassed by multiple distinct ranges in regions.gr. To use non-contiguous gene regions, provide gene_names in which some names are duplicated. For each unique gene in gene_names, this function will generate counts across all ranges for that gene, but be aware that it will only keep the largest range for each gene in the resulting DESeqDataSet object's rowRanges.

### A note on DESeq2 sizeFactors

DESeq2 sizeFactors are sample-specific normalization factors that are applied by division, i.e. $counts_{norm,i} = counts_i / sizeFactor_i$. This is in contrast to normalization factors as defined in this package (and commonly elsewhere), which are applied by multiplication. Also note that DESeq2's "normalizationFactors" are not sample specific, but rather gene specific factors used to correct for ascertainment bias across different genes (e.g. as might be relevant for GSEA or Go analysis).

### Author(s)

Mike DeBerardine

### See Also

DESeq2::DESeqDataSet, getDESeqResults, getDESeqResultsInBatch

---

## getDESeqResults *Get DESeq2 results using reduced dispersion matrices*

---

### Description

This function calls DESeq2::DESeq and DESeq2::results on a pre-existing DESeqDataSet object and returns a DESeqResults table for one or more pairwise comparisons. However, unlike a standard call to DESeq2::results using the contrast argument, this function subsets the dataset so that DESeq2 only estimates dispersion for the samples being compared, and not for all samples present.

### Usage

```
getDESeqResults(
  dds,
  contrast.numer,
  contrast.denom,
  comparisons.list = NULL,
  sizeFactors = NULL,
```

```
  alpha = 0.1,
  args.DESeq = NULL,
  args.results = NULL,
  ncores = detectCores(),
  quiet = FALSE
)
```

## Arguments

| | |
|---|---|
| dds | A DESeqDataSet object, produced using either getDESeqDataSet from this package or DESeqDataSet from DESeq2. If dds was not created using getDESeqDataSet, dds must be made with design = ~condition such that a unique condition level exists for each sample/treatment condition. |
| contrast.numer | A string naming the condition to use as the numerator in the DESeq2 comparison, typically the perturbative condition. |
| contrast.denom | A string naming the condition to use as the denominator in the DESeq2 comparison, typically the control condition. |
| comparisons.list | As an optional alternative to supplying a single contrast.numer and contrast.denom, users can supply a list of character vectors containing numerator-denominator pairs, e.g. list(c("B","A"),c("C","A"),c("C","B")). |
| sizeFactors | A vector containing DESeq2 sizeFactors to apply to each sample. Each sample's readcounts are *divided* by its respective DESeq2 sizeFactor. A warning will be generated if the DESeqDataSet already contains sizeFactors, and the previous sizeFactors will be over-written. |
| alpha | The significance threshold passed to DESeqResults. This won't affect the output results, but is used as a performance optimization by DESeq2. |
| args.DESeq | Additional arguments passed to DESeq, given as a list of argument-value pairs, e.g. list(test = "LRT",fitType = "local"). All arguments given here will be passed to DESeq except for object and parallel. If no arguments are given, all defaults will be used. |
| args.results | Additional arguments passed to DESeq2::results, given as a list of argument-value pairs, e.g. list(altHypothesis = "greater",lfcThreshold = 1.5). All arguments given here will be passed to results except for object, contrast, alpha, and parallel. If no arguments are given, all defaults will be used. |
| ncores | The number of cores to use for parallel processing. Multicore processing is only used if more than one comparison is being made (i.e. argument comparisons.list is used), and the number of cores utilized will not be greater than the number of comparisons being performed. |
| quiet | If TRUE, all output messages from calls to DESeq and results will be suppressed, although passing option quiet in args.DESeq will supersede this option for the call to DESeq. |

## Value

For a single comparison, the output is the DESeqResults result table. If a comparisons.list is used to make multiple comparisons, the output is a named list of DESeqResults objects, with elements named following the pattern "X_vs_Y", where X is the name of the numerator condition, and Y is the name of the denominator condition.

### Author(s)

Mike DeBerardine

### See Also

getDESeqDataSet, getDESeqResultsInBatch, DESeq2::results

---

`getDESeqResultsInBatch` *Automate batch calls to getDESeqResults*

---

### Description

This function can automate the generation numerous pairwise DESeq2 comparisons using several logical schemes.

### Usage

```
getDESeqResultsInBatch(
  dds,
  sizeFactors = NULL,
  alpha = 0.05,
  anchor = NULL,
  permutations = FALSE,
  additional_comparisons = NULL,
  ncores = detectCores()
)
```

### Arguments

`ncores`

---

`getMaxPositionsBySignal` *Find sites with max signal in regions of interest*

---

### Description

For each signal-containing region of interest, find the single site with the most signal. Sites can be found at base-pair resolution, or defined for larger bins.

### Usage

```
getMaxPositionsBySignal(
  regions.gr,
  dataset.gr,
  binsize = 1,
  bin.centers = FALSE,
  field = "score",
  keep.score = FALSE
)
```

### Arguments

| | |
|---|---|
| regions.gr | A GRanges object containing regions of interest. |
| dataset.gr | A GRanges object in which signal is contained in metadata (typically in the "score" field). |
| binsize | The size of bin in which to calculate signal scores. |
| bin.centers | Logical indicating if the centers of bins are returned, as opposed to the entire bin. If TRUE, |
| field | The metadata field of dataset.gr to be counted. |
| keep.score | Logical indicating if the signal value at the max site should be reported. If set to TRUE, the values are kept as a new metadata column in regions.gr. |

### Value

Output is a GRanges object with regions.gr metadata, but each range only contains the site within each regions.gr range that had the most signal. If binsize > 1, the entire bin is returned, unless bin.centers = TRUE, in which case a single-base site is returned. The site is set to the center of the bin, and if the binsize is even, the site is rounded to be closer to the beginning of the range.

If keep.score = TRUE, the output will also contain metadata for the signal at the max site. The output is *not* necessarily same length as regions.gr, as regions without signal are not returned. If *no regions* have signal (e.g. as could happen if running this function on a single region), the function will return an empty GRanges object with intact metadata columns.

### Author(s)

Mike DeBerardine

### See Also

getCountsByPositions

---

## getPausingIndices *Calculate pausing indices from user-supplied promoters & genebodies*

---

### Description

Pausing index (PI) is calculated for each gene (within matched promoters.gr and genebodies.gr) as promoter-proximal (or pause region) signal counts divided by genebody signal counts. If length.normalize = TRUE (recommended), the signal counts within each range in promoters.gr and genebodies.gr are divided by their respective range widths (region lengths) before pausing indices are calculated.

### Usage

```
getPausingIndices(
  dataset.gr,
  promoters.gr,
  genebodies.gr,
  field = "score",
  length.normalize = TRUE,
  remove.empty = FALSE,
  ncores = detectCores()
)
```

### Arguments

| | |
|---|---|
| dataset.gr | A GRanges object in which signal is contained in metadata (typically in the "score" field). |
| promoters.gr | A GRanges object containing promoter-proximal regions of interest. |
| genebodies.gr | A GRanges object containing genebody regions of interest. |
| field | The metadata field of dataset.gr to be counted. If length(field) > 1, a dataframe is returned containing the pausing indices for each region in each field. |
| length.normalize | A logical indicating if signal counts within regions of interest should be length normalized. The default is TRUE, which is recommended, especially if input regions don't all have the same width. |
| remove.empty | A logical indicating if genes without any signal in promoters.gr should be removed. No genes are filtered by default. |
| ncores | Multiple cores can only be used if length(field) > 1. |

### Value

A vector of length given by the length of the genelist (or possibly shorter if remove.empty = TRUE). If length(field) > 1, a dataframe is returned, containing a column for each field.

### Author(s)

Mike DeBerardine

### See Also

getCountsByRegions

---

## getStrandedCoverage *Get strand-specific coverage*

---

### Description

Computes strand-specific coverage signal, and returns a GRanges object with signal in the "score" metadata column. Function also works for non-strand-specific data. Note that output is not automatically converted into a "basepair-resolution" GRanges object.

### Usage

```
getStrandedCoverage(dataset.gr, field = "score")
```

### Arguments

| | |
|---|---|
| dataset.gr | A GRanges object either containing ranges for each read, or one in which read-counts for individual ranges are contained in metadata (typically in the "score" field). |
| field | The name of the metadata field that contains readcounts. If no metadata field contains readcounts, and each range represents a single read, set to NULL. |

## Author(s)

Mike DeBerardine

## See Also

makeGRangesBPres

---

`import-functions` *Import basepair-resolution files*

---

## Description

Import basepair-resolution files

## Usage

```
import_bigWig(
  plus_file,
  minus_file,
  genome = NULL,
  keep.X = TRUE,
  keep.Y = TRUE,
  keep.M = FALSE,
  keep.nonstandard = FALSE
)

import_bedGraph(
  plus_file,
  minus_file,
  genome = NULL,
  keep.X = TRUE,
  keep.Y = TRUE,
  keep.M = FALSE,
  keep.nonstandard = FALSE
)
```

## Arguments

`plus_file, minus_file`
: Paths for strand-specific input files.

`genome`
: Optional string for UCSC reference genome, e.g. "hg38". If given, non-standard chromosomes are trimmed, and options for sex and mitochondrial chromosomes are applied.

`keep.X, keep.Y, keep.M, keep.nonstandard`
: Logicals indicating which non-autosomes should be kept. By default, sex chromosomes are kept, but mitochondrial and non-standard chromosomes are removed.

### Details

Imports a GRanges object containing base-pair resolution data, with the `score` metadata column indicating the number of reads represented by each range.

`import_bedGraph` is useful for when both 5'- and 3'-end information is to be maintained for each sequenced molecule. It effectively imports the entire read.

For `import_bigWig`, all ranges are of width = 1.

### Author(s)

Mike DeBerardine

### See Also

`tidyChromosomes`, `rtracklayer::import`

---

## `makeGRangesBPres` *Make base-pair resolution GRanges object*

---

### Description

Splits up all ranges in `gr` to be each 1 basepair wide. All information is preserved, including all metadata. To wit, `length(output.gr)` = `sum(width(dataset.gr))`.

### Usage

```
makeGRangesBPres(dataset.gr)
```

### Arguments

| | |
|---|---|
| `dataset.gr` | A disjoint GRanges object |

### Details

Note that this function doesn't perform any transformation on the metadata in the input; for any ranges of width > 1, the metadata is simply copied to the daughters of that range (whose widths are all equal to 1).

This function is intended to work on datasets at single-base resolution. Data of this type is often formatted as a bigWig file, and any data imported from a bigWig file by rtracklayer is suitable for processing. bigWig files will typically use run-length compression on the data signal (the 'score' column), such that when imported by rtracklayer, adjacent bases sharing the same signal will combined into a single range. The base-pair resolution GRanges objects produced by this function remove this compression, resulting in each index (each range) of the GRanges object addressing a single genomic position.

To properly use base-pair resolution information, the user should be selecting a single-base from each read, which can be accomplished using `GenomicRanges::resize()`. Then, single-base coverage can be calculated using `getStrandedCoverage`.

### Author(s)

Mike DeBerardine

## See Also

getStrandedCoverage, GenomicRanges::resize()

---

`mergeGRangesData` *Merge base-pair resolution GRanges objects*

---

## Description

Merges 2 or more GRanges objects. For each object, the range widths must all be 1, and the `score` metadata column contains coverage information at each site. This function returns a single GRange object containing all sites of the input objects, and the sum of all scores at all sites.

## Usage

```
mergeGRangesData(..., field = "score", ncores = detectCores())
```

## Arguments

| | |
|---|---|
| `...` | Any number of GRanges objects in which signal (e.g. readcounts) are contained within metadata. |
| `field` | One or more metadata fields to be combined, typically the "score" field. Fields typically contain coverage information. |
| `ncores` | More than one core can be used to coerce non-single-width GRanges objects using `makeGRangesBPres`. |

## Author(s)

Mike DeBerardine

## See Also

makeGRangesBPres

---

`metaSubsample` *Iterative Subsampling for Metaplotting*

---

## Description

This function performs bootstrap subsampling of mean readcounts at different positions within regions of interest. Mean signal counts can be estimated at base-pair resolution, or smoothed over larger bins.

## Usage

```
metaSubsample(
  dataset.gr,
  regions.gr,
  binsize = 1,
  first.output.xval = 1,
  sample.name = deparse(substitute(dataset.gr)),
  n.iter = 1000,
  prop.sample = 0.1,
  lower = 0.125,
  upper = 0.875,
  NF = 1,
  field = "score",
  remove.empty = FALSE,
  ncores = 1
)
```

## Arguments

| | |
|---|---|
| `dataset.gr` | A GRanges object in which signal is contained in metadata (typically in the `"score"` field). |
| `regions.gr` | A GRanges object containing intervals over which to metaplot. All ranges must have the same width. |
| `binsize` | The size of bin (number of columns, e.g. basepairs) to use for metaplotting. Especially important for metaplots over large/sparse regions. |
| `first.output.xval` | The relative start position of the first bin, e.g. if regions.gr begins at 50 bases upstream of the TSS, set `first.output.xval = -50`. This number only affects the x-values that are returned, which are provided as a convenience. |
| `sample.name` | Defaults to the name of `dataset.gr`. This is included in the output as a convenience for row-binding outputs from different samples. |
| `n.iter` | Number of random subsampling iterations to perform. Default is `1000`. |
| `prop.sample` | The proportion of the ranges in `regions.gr` (e.g. the proportion of genes) to subsample in each iteration. The default is `0.1` (10 percent). |
| `lower` | The lower quantile of subsampled signal means to return. The default is `0.125` (12.5th percentile). |
| `upper` | The upper quantile of subsampled signal means to return. The default is `0.875` (85.5th percentile). |
| `NF` | Optional normalization factor by which to multiply the counts. |
| `field` | The metadata field of `dataset.gr` to be counted. |
| `remove.empty` | A logical indicating whether regions without signal should be removed from the analysis. |
| `ncores` | Number of cores to use for parallel computation. No parallel processing is used by default, as there's no performance benefit for typical usage with short computation times. |

## Value

Dataframe containing x-values, means, lower quantiles, upper quantiles, and the sample name (as a convenience for row-binding multiple of these dataframes).

## Author(s)

Mike DeBerardine

## See Also

metaSubsampleMatrix, getCountsByPositions

---

`metaSubsampleMatrix` *Iterative Subsampling for Metaplotting (On Count Matrices)*

---

## Description

In the most general sense, this function performs iterations of randomly subsampling rows of a matrix, and returns a summary of mean values calculated for each column. The typical application is for generating metaplots, with the typical input being a matrix in which each row is a gene or other region of interest, each column is a position within that gene (either a specific basepair or a bin), and element values are signal (e.g. read counts) within those positions.

## Usage

```
metaSubsampleMatrix(
  counts.mat,
  binsize = 1,
  first.output.xval = 1,
  sample.name = deparse(substitute(counts.mat)),
  n.iter = 1000,
  prop.sample = 0.1,
  lower = 0.125,
  upper = 0.875,
  NF = 1,
  ncores = 1
)
```

## Arguments

| | |
|---|---|
| `counts.mat` | A matrix of signal counts in which rows are regions of interest and columns are sites/bins in each region. |
| `binsize` | The size of bin (number of columns, e.g. basepairs) to use for metaplotting. Especially important for metaplots over large/sparse regions. |
| `first.output.xval` | The relative start position of the first bin, e.g. if regions.gr begins at 50 bases upstream of the TSS, set `first.output.xval = -50`. This number only affects the x-values that are returned, which are provided as a convenience. |
| `sample.name` | Defaults to the name of `dataset.gr`. |
| `n.iter` | Number of random subsampling iterations to perform. Default is 1000. |
| `prop.sample` | The proportion of rows to subsample in each iteration. The default is 0.1. |
| `lower` | The lower quantile of subsampled signal means to return. The default is 0.125 (12.5th percentile). |

| | |
|---|---|
| upper | The upper quantile of subsampled signal means to return. The default is 0.875 (85.5th percentile). |
| NF | Optional normalization factor by which to multiply the counts. |
| ncores | Number of cores to use for parallel computation. As of writing, parallel processing doesn't show any benefit for short computation times (e.g. <1 minute for our typical experience on a laptop). |

## Value

Dataframe containing x-values, means, lower quantiles, upper quantiles, and the sample name (as a convenience for row-binding multiple of these dataframes).

## Author(s)

Mike DeBerardine

## See Also

metaSubsample, getCountsByPositions

---

PROseq *PRO-seq data from Drosophila S2 cells*

---

## Description

PRO-seq data of Drosophila S2 cells, chromosome 4.

## Usage

```
PROseq
```

## Format

A disjoint GRanges object with 47533 ranges with 1 metadata column:

**score** coverage of PRO-seq read 3'-ends ...

## Details

Hojoong Kwak, Nicholas J. Fuda, Leighton J. Core, John T. Lis (2013). Precise Maps of RNA Polymerase Reveal How Promoters Direct Initiation and Pausing. Science, 339(6122), 950–953. https://doi.org/10.1126/science.1229386

## Source

GEO Accession GSM1032758, run SRR611828.

## PROseq_paired *Paired PRO-seq data from Drosophila S2 cells*

### Description

PRO-seq data of Drosophila S2 cells, chromosome 4. Entire mapped reads kept.

### Usage

```
PROseq_paired
```

### Format

A GRanges object with 52464 ranges with 1 metadata column:

**score** number of reads sharing the same mapped 5' and 3' ends ...

### Details

Hojoong Kwak, Nicholas J. Fuda, Leighton J. Core, John T. Lis (2013). Precise Maps of RNA Polymerase Reveal How Promoters Direct Initiation and Pausing. Science, 339(6122), 950–953. https://doi.org/10.1126/science.1229386

### Source

GEO Accession GSM1032758, run SRR611828.

## subsampleGRanges *Randomly subsample reads from GRanges dataset*

### Description

Random subsampling is not performed on ranges, but on reads. Readcounts should be given as a metadata field (usually "score"), and should normally be integers. If normalized readcounts are given, an attempt will be made to infer the normalization factor based on the least-common-multiple of the signal found in the specified field. This function can also subsample ranges directly if `field = NULL`, but the `sample` function can be used in this scenario.

### Usage

```
subsampleGRanges(dataset.gr, n = NULL, prop = NULL, field = "score")
```

### Arguments

| | |
|---|---|
| `dataset.gr` | A GRanges object in which signal (e.g. readcounts) are contained within metadata. |
| `n` | Number of reads to subsample. Either `n` or `prop` can be given. |
| `prop` | Proportion of total signal to subsample. |
| `field` | The metadata field of `dataset.gr` that contains readcounts for reach position. If each range represents a single read, set `field = NULL` |

### Author(s)

Mike DeBerardine

---

`subsetRegionsBySignal` *Subset regions of interest by quantiles of overlapping signal*

---

### Description

A convenience function to subset regions of interest by the amount of signal they contain, according to their quantile (i.e. their signal ranks).

### Usage

```
subsetRegionsBySignal(
  regions.gr,
  dataset.gr,
  quantiles = c(0.5, 1),
  field = "score",
  order.by.rank = FALSE,
  density = FALSE
)
```

### Arguments

| | |
|---|---|
| `regions.gr` | A GRanges object containing regions of interest. |
| `dataset.gr` | A GRanges object in which signal is contained in metadata (typically in the "score" field). |
| `quantiles` | A value pair giving the lower quantile and upper quantile of regions to keep. Regions with signal quantiles below than the lower quantile are removed, while regions with signal quantiles above the upper quantile are removed. Quantiles must be in range `(0,1)`. An empty GRanges object is returned if `lower quantile = 1` or `upper quantile = 0`. |
| `field` | The metadata field of `dataset.gr` to be counted. |
| `order.by.rank` | If `TRUE`, the output regions are sorted based on the amount of signal contained (in decreasing order). If `FALSE` (the default), genes are sorted by their positions. |
| `density` | A logical indicating whether signal counts should be normalized to the width of ranges in `regions.gr`. By default, the function only considers the total signal in each range. |

### Details

Typical uses may include removing the 5 signal (`lower_quantile = 0.05`) and the 5 (`upper_quantile = 0.95`), or returning the middle 50 signal (`lower_quantile = 0.25`, `upper_quantile = 0.75`). If `lower_quantile = 0` and `upper_quantile = 1`, all regions are returned, but the returned regions will be sorted by position, or by score if `order.by.rank = TRUE`.

### Value

A GRanges object of length `length(regions.gr) * (upper_quantile -lower_quantile)`.

### Author(s)

Mike DeBerardine

### See Also

getCountsByRegion

---

tidyChromosomes *Remove odd chromosomes from GRanges objects*

---

### Description

This convenience function removes non-standard, mitochondrial, and/or sex chromosomes from any GRanges object. For the chromosomes being removed, any ranges found on those chromosomes are removed, and the chromosomes are also removed from seqinfo. Standard chromosomes are defined using the standardChromosomes function from the GenomeInfoDb package.

### Usage

```
tidyChromosomes(
  gr,
  keep.X = TRUE,
  keep.Y = TRUE,
  keep.M = FALSE,
  keep.nonstandard = FALSE
)
```

### Arguments

| | |
|---|---|
| gr | Any GRanges object, however the object should have a standard genome set, e.g. genome(gr) <-"hg38" |
| keep.X, keep.Y, keep.M, keep.nonstandard | Logicals indicating which non-autosomes should be kept. By default, sex chromosomes are kept, but mitochondrial and non-standard chromosomes are removed. |

### Author(s)

Mike DeBerardine

### See Also

GenomeInfoDb::standardChromosomes,

`txs_dm6_chr4` *Ensembl transcripts for Drosophila melanogaster, dm6, chromosome 4.*

## Description

Transcripts obtained from annotation package TxDb.Dmelanogaster.UCSC.dm6.ensGene, which was in turn made by the Bioconductor Core Team from UCSC resources on 2019-04-25. Metadata columns were obtained from "TXNAME" and "GENEID" columns. Data exported from the TxDb package using GenomicFeatures version 1.35.11 on 2019-12-19.

## Usage

```
txs_dm6_chr4
```

## Format

A GRanges object with 339 ranges and 2 metadata columns:

**tx_name** Flybase unique identifiers for transcripts

**gene_id** FLybase unique identifiers for the associated genes

## Source

TxDb.Dmelanogaster.UCSC.dm6.ensGene version 3.4.6

# Index