Package 'seqsetvis'

April 15, 2020

Type Package

Title Set Based Visualizations for Next-Gen Sequencing Data

Version 1.6.0

Description seqsetvis enables the visualization and analysis of multiple genomic samples. Although seqsetvis was designed for the comparison of mulitple ChIP-seq samples, this package is domain-agnostic and allows the processing of multiple genomic coordinate files (bed-like files) and signal files (bigwig files or bam pileups).

License MIT + file LICENSE

Encoding UTF-8

LazyData true

Suggests BiocFileCache, BiocManager, BiocStyle, ChIPpeakAnno, covr, cowplot, knitr, rmarkdown, testthat

Depends R (>= 3.5), ggplot2

Imports data.table, eulerr, GenomeInfoDb, GenomicAlignments, GenomicRanges, grDevices, grid, IRanges, limma, methods, parallel, pbapply, png, RColorBrewer, Rsamtools, rtracklayer, S4Vectors, stats

RoxygenNote 6.1.1

VignetteBuilder knitr

NeedsCompilation no

biocViews Software, ChIPSeq, MultipleComparison, Sequencing, Visualization

git_url https://git.bioconductor.org/packages/seqsetvis

git_branch RELEASE_3_10

git_last_commit 17b5f4f

git_last_commit_date 2019-10-29

Date/Publication 2020-04-14

Author Joseph R Boyd [aut, cre]

Maintainer Joseph R Boyd <jrboyd@uvm.edu>

R topics documented:

`seqsetvis-package` *easy awesome peak set vis TESTING seqsetvis allows you to...*

Description

2 steps `ssvOverlapIntervalSets`. `ssvFetchBigwig`. Otherwise refer to the vignettes to see

Author(s)

Maintainer: Joseph R Boyd <jrboyd@uvm.edu>

`.expand_cigar_dt` *Expand intermediate bam fetch by cigar codes*

Description

see sam specs for cigar details

Usage

```
.expand_cigar_dt(cigar_dt, op_2count = c("M", "D", "=", "X"))
```

Arguments

`cigar_dt`	data.table with 5 required named columns in any order. c("which_label", "seqnames", "strand", "start", "cigar")
`op_2count`	Cigar codes to count. Default is alignment (M), deletion (D), match (=), and mismatch (X). Other useful codes may be skipped regions for RNA splicing (N). The locations of any insterions (I) or clipping/padding (S, H, or P) will be a single bp immediately before the interval.

Value

data.table with cigar entries expanded

`.expand_cigar_dt_recursive` *Expand intermediate bam fetch by cigar codes*

Description

see sam specs for cigar details

Usage

```
.expand_cigar_dt_recursive(cigar_dt)
```

Arguments

`cigar_dt`	data.table with 5 required named columns in any order. c("which_label", "seqnames", "strand", "start", "cigar")

Value

data.table with cigar entries expanded

`.rm_dupes` *Remove duplicate reads based on stranded start position. This is an over-simplification. For better duplicate handling, duplicates must be marked in bam and flag passed to fetchBam() ... for ScanBamParam*

Description

flag = scanBamFlag(isDuplicate = FALSE)

Usage

```
.rm_dupes(reads_dt, max_dupes)
```

Arguments

`reads_dt`	data.table of reads as loaded by fetchBam
`max_dupes`	maximum allowed positional duplicates

Value

reads_dt with duplicated reads over max_dupes removed

`.rm_dupesPE` *Remove duplicate paired-end reads based on start and end position. This is an over-simplification. For better duplicate handling, duplicates must be marked in bam and flag passed to fetchBamPE() ... for ScanBamParam*

Description

flag = scanBamFlag(isDuplicate = FALSE)

Usage

```
.rm_dupesPE(reads_dt, max_dupes)
```

Arguments

`reads_dt`	data.table of reads as loaded by fetchBamPE
`max_dupes`	maximum allowed positional duplicates

Value

reads_dt with duplicated reads over max_dupes removed

`append_ynorm` *append_ynorm*

Description

see `calc_norm_factors` for normalization details.

Usage

```
append_ynorm(full_dt, value_ = "y", cap_value_ = "y_cap_value",
  norm_value_ = "y_norm", by1 = "id", by2 = "sample",
  aggFUN1 = max, aggFUN2 = function(x) quantile(x, 0.95),
  cap_dt = NULL, do_not_cap = FALSE)
```

Arguments

`full_dt`	a data.table, as returned by ssvFetch*(..., return_data.table = TRUE).
`value_`	character, attribute in full_dt to normalzie.
`cap_value_`	character, new attribute name specifying values to cap to.
`norm_value_`	character, new attribute name specifying normalized values.
`by1`	character vector, specifies attributes relevant to step 1.
`by2`	character vector, specifies attributes relevant to step 1 and 2.
`aggFUN1`	function called on value_ with by = c(by1, by2) in step 1.
`aggFUN2`	function called on result of aggFUN1 with by = by2 in step 2.
`cap_dt`	optionally, provide user generated by2 to cap_value_ mapping
`do_not_cap`	if TRUE, normalized values are not capped to 1. Default is FALSE.

Value

data.table, full_dt with cap_value_ and norm_value_ values appended.

Examples

```
append_ynorm(CTCF_in_10a_profiles_dt)
append_ynorm(CTCF_in_10a_profiles_dt,
  aggFUN1 = mean, aggFUN2 = function(x)quantile(x, .5))
```

`applySpline` *applies a spline smoothing to a tidy data.table containing x and y values.*

Description

`applySpline` Is intended for two-dimensional tidy data.tables, as retured by `ssvFetchBigwig`

Usage

```
applySpline(dt, n, x_ = "x", y_ = "y", by_ = "",
  splineFun = stats::spline)
```

Arguments

`dt`	a tidy data.table containing two-dimensional data
`n`	the number of interpolation points to use per input point, see `?spline`. n must be > 1.
`x_`	the variable name of the x-values
`y_`	the variable name of the y-values
`by_`	optionally, any variables that provide grouping to the data. default is none. see details.
`splineFun`	a function that accepts x, y, and n as arguments and returns a list of length 2 with named elements x and y. `stats::spline` by default. see `stats::spline` for details.

Details

by_ is quite powerful. If `by_ = c('gene_id','sample_id')`, splines will be calculated individually for each gene in each sample. alternatively if `by_ = c('gene_id')`

Value

a newly derived data.table that is `n` times longer than original.

See Also

`ssvFetchBigwig`

Examples

```
#data may be blockier than we'd like
ggplot(CTCF_in_10a_profiles_dt[, list(y = mean(y)), by = list(sample, x)]) +
    geom_line(aes(x = x, y = y, color = sample))

#can be smoothed by applying a spline  (think twice about doing so,
#it may look prettier but may also be deceptive or misleading)

splined_smooth = applySpline(CTCF_in_10a_profiles_dt, n = 10,
    y_ = 'y', by_ = c('id', 'sample'))
ggplot(splined_smooth[, list(y = mean(y)), by = list(sample, x)]) +
    geom_line(aes(x = x, y = y, color = sample))
```

Bcell_peaks *4 random peaks for paired-end data*

Description

matches `system.file("extdata/Bcell_PE.mm10.bam",package = "seqsetvis")`

Format

GRanges length 4

Details

this is included only for testing ssvFetchBamPE functions.

---

calc_norm_factors *calc_norm_factors*

Description

Calculate normalization factors in a two step process:

Usage

```
calc_norm_factors(full_dt, value_ = "y", cap_value_ = "y_cap_value",
  by1 = "id", by2 = "sample", aggFUN1 = max, aggFUN2 = function(x)
  quantile(x, 0.95))
```

Arguments

full_dt	a data.table, as returned by ssvFetch*(..., return_data.table. = TRUE)
value_	character, attribute in full_dt to normalzie.
cap_value_	character, new attribute name specifying values to cap to.
by1	character vector, specifies attributes relevant to step 1.
by2	character vector, specifies attributes relevant to step 1 and 2.
aggFUN1	function called on value_ with by = c(by1, by2) in step 1.
aggFUN2	function called on result of aggFUN1 with by = by2 in step 2.

Details

1) summarize every region for each sample (default summary function is max)

2) caclulate a value to cap each sample to based on regions (default is 95th quantile).

The uderlying assumption here is that meaningful enrichment is present at the majority of regions provided. If prevalence varies by a specific factor, say ChIP-seq targets with different characteristics - ie. when analyzing TSSes for H3K4me3 and an infrequent transcription factor it is more appropriate to specify appropriate quantile cutoffs per factor.

Value

data.table mapping by2 to cap_value_.

Examples

```
calc_norm_factors(CTCF_in_10a_profiles_dt)
calc_norm_factors(CTCF_in_10a_profiles_dt,
  aggFUN1 = mean, aggFUN2 = function(x)quantile(x, .5))
```

---

centerAtMax — *centers profile of x and y. default is to center by region but across all samples.*

Description

`centerAtMax` locates the coordinate x of the maximum in y and shifts x such that it is zero at max y.

Usage

```
centerAtMax(dt, x_ = "x", y_ = "y", by_ = "id", view_size = NULL,
  trim_to_valid = TRUE, check_by_dupes = TRUE, x_precision = 3,
  replace_x = TRUE)
```

Arguments

`dt`	data.table
`x_`	the variable name of the x-values. default is 'x'
`y_`	the variable name of the y-values default is 'y'
`by_`	optionally, any variables that provide grouping to the data. default is none. see details.
`view_size`	the size in `x_` to consider for finding the max of `y_`. if length(view_size) == 1, range will be c(-view_size, view_size). if length(view_size) > 1, range will be range(view_size). default value of NULL uses complete range of x.
`trim_to_valid`	valid `x_` values are those with a set `y_` value in all `by_` combinations
`check_by_dupes`	default assumption is that there should be on set of x_ for a by_ instance. if this is not the case and you want to disable warnings about set this to FALSE.
`x_precision`	numerical precision of x, default is 3.
`replace_x`	logical, default TRUE. if TRUE x_ will be replaced with position relative to summit. if FALSE x_ will be preserved and x_summitPosition added.

Details

character. by_ controls at the level of the data centering is applied. If by_ is "" or NULL, a single max position will be determined for the entire dataset. If by is "id" (the default) then each region will be centered individually across all samples.

Value

data.table with x (or xnew if replace_x is FALSE) shifted such that x = 0 matches the maximum y-value define by by_ grouping

Examples

```
centerAtMax(CTCF_in_10a_profiles_gr, y_ = 'y', by_ = 'id',
  check_by_dupes = FALSE)
#it's a bit clearer what's happening with trimming disabled
#but results are less useful for heatmaps etc.
centerAtMax(CTCF_in_10a_profiles_gr, y_ = 'y', by_ = 'id',
  check_by_dupes = FALSE, trim_to_valid = FALSE)
#specify view_size to limit range of x values considered, prevents
#excessive data trimming.
centerAtMax(CTCF_in_10a_profiles_gr, y_ = 'y', view_size = 100, by_ = 'id',
check_by_dupes = FALSE)
```

---

centerFixedSizeGRanges

Transforms set of GRanges to all have the same size.

Description

`centerFixedSizeGRanges` First calculates the central coordinate of each GRange in `grs` and extends in both direction by half of `fixed_size`

Usage

```
centerFixedSizeGRanges(grs, fixed_size = 2000)
```

Arguments

`grs`	Set of GRanges with incosistent and/or incorrect size
`fixed_size`	The final width of each GRange returned.

Value

Set of GRanges after resizing all input GRanges, either shortened or lengthened as required to match `fixed_size`

Examples

```
library(GenomicRanges)
grs = GRanges("chr1", IRanges(1:10+100, 1:10*3+100))
centered_grs = centerFixedSizeGRanges(grs, 10)
width(centered_grs)
```

`chromHMM_demo_bw_states_gr` *MCF10A CTCF profiles at 20 windows per chromHMM state, hg38.*

Description

MCF10A CTCF profiles at 20 windows per chromHMM state, hg38.

Format

a GRanges object of length 4000 with 5 metadata columns sufficient for use with ggplot2

Details

part of `chromHMM_demo_data`

the result of ssvFetchBigwig() on the MCF10A_CTCF_FE.bw near 20 randomly selected windows per chromHMM state.

`chromHMM_demo_chain_url` *URL to download hg19ToHg38 liftover chain from UCSC*

Description

URL to download hg19ToHg38 liftover chain from UCSC

Format

a character containing a URL

Details

file is gzipped .txt

part of `chromHMM_demo_data`

`chromHMM_demo_data` *chromHMM state segmentation in the MCF7 cell line*

Description

Vignette data for seqsetvis was downloaded directly from GEO series GSE57498. This data is the state segmentation by chromHMM in the MCF7 cell line. chromHMM creates a hidden markov model by integrating several ChIP-seq samples, in this case:

- MCF7_H3K27ac_ChIP-Seq
- MCF7_H3K27me3_ChIP-Seq
- MCF7_H3K4me1_ChIP-Seq
- MCF7_H3K4me3_ChIP-Seq
- MCF7_RNApolIIp_ChIP-Seq

Data from GEO series GSE57498 is from the publication Taberlay PC et al. 2014

Details

Contains:

- `chromHMM_demo_overlaps_gr`
- `chromHMM_demo_bw_states_gr`
- `chromHMM_demo_state_total_widths`
- `chromHMM_demo_state_colors`
- `chromHMM_demo_segmentation_url`
- `chromHMM_demo_chain_url`

`chromHMM_demo_overlaps_gr` *overlap of MCF10A CTCF with MCF7 chromHMM states, hg38.*

Description

overlap of MCF10A CTCF with MCF7 chromHMM states, hg38.

Format

a GRanges object of length 98 with 10 logical metadata columns, 1 per state.

Details

part of `chromHMM_demo_data`

the result of ssvOverlapIntervalSets() on MCF10A CTCF peaks and MCF7 chromHMM states with use_first = TRUE

first (the MCF10A peaks) and no_hit columns have been removed each remaining column represents MCF10A peaks overlapping with a state.

`chromHMM_demo_segmentation_url`
URL to download hg19 MCF7 chromHMM segmentation

Description

URL to download hg19 MCF7 chromHMM segmentation

Format

a character containing a URL

Details

file is gzipped bed with name, score, itemRgb and thick meta columns

part of `chromHMM_demo_data`

`chromHMM_demo_state_colors`
original state name to color mappings stored in segmentation bed

---

Description

original state name to color mappings stored in segmentation bed

Format

a named character vector mapping states to hex colors

Details

part of `chromHMM_demo_data`

`chromHMM_demo_state_total_widths`
state name to total width mappings, hg38

Description

state name to total width mappings, hg38

Format

named numeric of total widths per state

Details

part of `chromHMM_demo_data`

`clusteringKmeans` *perform kmeans clustering on matrix rows and return reordered matrix along with order matched cluster assignments. clusters are sorted using hclust on centers*

Description

perform kmeans clustering on matrix rows and return reordered matrix along with order matched cluster assignments. clusters are sorted using hclust on centers

Usage

```
clusteringKmeans(mat, nclust, seed = NULL)
```

Arguments

`mat`	numeric matrix to cluster
`nclust`	the number of clusters
`seed`	DEPRECATED. Call set.seed() prior to this funciton to allow reproducibility.

Value

data.table with group variable indicating cluster membership and id variable that is a factor indicating order based on within cluster similarity

Examples

```
dt = data.table::copy(CTCF_in_10a_profiles_dt)
mat = data.table::dcast(dt, id ~ sample + x, value.var = "y" )
rn = mat$id
mat = as.matrix(mat[,-1])
rownames(mat) = rn
clust_dt = clusteringKmeans(mat, nclust = 3)
dt = merge(dt, clust_dt)
dt$id = factor(dt$id, levels = clust_dt$id)
dt[order(id)]
```

`clusteringKmeansNestedHclust` *perform kmeans clustering on matrix rows and return reordered matrix along with order matched cluster assignments clusters are sorted using hclust on centers the contents of each cluster are sorted using hclust*

Description

perform kmeans clustering on matrix rows and return reordered matrix along with order matched cluster assignments clusters are sorted using hclust on centers the contents of each cluster are sorted using hclust

Usage

```
clusteringKmeansNestedHclust(mat, nclust, seed = NULL)
```

Arguments

mat	A wide format matrix
nclust	the number of clusters
seed	passed to set.seed() to allow reproducibility

Value

data.table with 2 columns of cluster info. id column corresponds with input matrix rownames and is sorted within each cluster using hierarchical clusering group column indicates cluster assignment

Examples

```
dt = data.table::copy(CTCF_in_10a_profiles_dt)
mat = data.table::dcast(dt, id ~ sample + x, value.var = "y" )
rn = mat$id
mat = as.matrix(mat[,-1])
rownames(mat) = rn
clust_dt = clusteringKmeansNestedHclust(mat, nclust = 3)
dt = merge(dt, clust_dt)
dt$id = factor(dt$id, levels = clust_dt$id)
dt[order(id)]
```

---

col2hex *converts a valid r color name ("black", "red", "white", etc.) to a hex value*

Description

converts a valid r color name ("black", "red", "white", etc.) to a hex value

Usage

```
col2hex(color_name)
```

Arguments

color_name	character. one or more r color names.

Value

hex value of colors coded by colors()

Examples

```
col2hex(c("red", "green", "blue"))
col2hex(c("lightgray", "gray", "darkgray"))
```

crossCorrByRle *Calculate cross correlation by using shiftApply on read coverage Rle*

Description

Calculate cross correlation by using shiftApply on read coverage Rle

Usage

```
crossCorrByRle(bam_file, query_gr, max_dupes = 1,
  fragment_sizes = 50:300, read_length = NULL, flip_strand = FALSE,
  ...)
```

Arguments

bam_file	character. Path to .bam file, must have index at .bam.bai.
query_gr	GRanges. Regions to calculate cross correlation for.
max_dupes	integer. Duplicate reads above this value will be removed.
fragment_sizes	integer. fragment size range to search for maximum correlation.
read_length	integer. Any values outside fragment_range that must be searched. If not supplied will be determined from bam_file. Set as NA to disable this behavior.
flip_strand	boolean. if TRUE strands that reads align to are swapped. This is typically only necessary if there was a mismatch between library chemistry and aligner settings. Default is FALSE.
...	arguments passed to ScanBamParam

Value

named list of results

Examples

```
bam_f = system.file("extdata/test.bam",
    package = "seqsetvis", mustWork = TRUE)
query_gr = CTCF_in_10a_overlaps_gr[1:2]
crossCorrByRle(bam_f, query_gr[1:2], fragment_sizes = seq(50, 300, 50))
```

CTCF_in_10a_bigWig_urls *FTP URL path for vignette data.*

Description

FE bigWig tracks for CTCF ChIP-seq in a MCF10A progression model. See GEO series GSE98551 for details.

Format

named character vector of length 3

Details

part of CTCF_in_10a_data

CTCF_in_10a_data *CTCF ChIP-seq in breast cancer cell lines*

Description

Vignette data for seqsetvis was downloaded directly from GEO series GSE98551. This data is CTCF ChIP-seq from a model of breast cancer progression derived from the MCF10A cell line.

Data from GEO series GSE98551 is from the publication Fritz AJ et al. 2018

Details

Contains:

- CTCF_in_10a_overlaps_gr
- CTCF_in_10a_profiles_dt
- CTCF_in_10a_bigWig_urls
- CTCF_in_10a_narrowPeak_urls

CTCF_in_10a_narrowPeak_grs
list of GRanges that results in 100 random subset when overlapped

---

Description

list of GRanges that results in 100 random subset when overlapped

Format

named character vector of length 3

Details

part of CTCF_in_10a_data

CTCF_in_10a_narrowPeak_urls

FTP URL path for vignette data. from

Description

macs2 peak calls for CTCF ChIP-seq in a MCF10A progression model. See GEO series GSE98551 for details.

Format

named character vector of length 3

Details

part of CTCF_in_10a_data

CTCF_in_10a_overlaps_gr

100 randomly selected regions from overlapping CTCF peaks in 10a cell ChIP-seq

Description

MACS2 narrowPeak calls on pooled biological replicates at pval 1e-5 and then 0.05 IDR filtered. IDR cutoffs determined by comparing top 150,000 pvalue sorted peak in replicates.

Format

GenomicRanges with 3 metadata columns of membership table

Details

See GEO series GSE98551 for details.

part of CTCF_in_10a_data

`CTCF_in_10a_profiles_dt`

Profiles for 100 randomly selected regions from overlapping CTCF peaks in 10a cell ChIP-seq Results from fetching bigwigs with CTCF_in_10a_overlaps_gr.

Description

A tidy data.table at window size 50 bp within 350 bp of peak center The variables are as follows:

Format

A tidy data.table of 2100 rows and 9 columns

Details

part of `CTCF_in_10a_data`

1. seqnames. chromosome for GRanges compatibility
2. start. start of interval
3. end. end of interval
4. width. width of interval
5. strand. leftover from GRanges.
6. id. unique identifier
7. y. fold-enrichment over input.
8. x. bp relative to center
9. sample. name of originating sample

`CTCF_in_10a_profiles_gr`

Profiles for 100 randomly selected regions from overlapping CTCF peaks in 10a cell ChIP-seq Results from CTCF_in_10a_overlaps_gr

---

Description

A tidy GRanges at window size 50 bp within 350 bp of peak center The variables are as follows:

Format

A tidy GRanges of 2100 rows and 4 metadata columns

Details

part of `CTCF_in_10a_data`

1. id. unique identifier
2. y. fold-enrichment over input.
3. x. bp relative to center
4. sample. name of originating sample

easyLoad_bed *easyLoad_bed takes a character vector of file paths to bed plus files and returning named list of GRanges. Mainly a utility function for loading MACS2 narrowPeak and broadPeak.*

Description

easyLoad_bed takes a character vector of file paths to bed plus files and returning named list of GRanges. Mainly a utility function for loading MACS2 narrowPeak and broadPeak.

Usage

```
easyLoad_bed(file_paths, file_names = NULL, extraCols = character())
```

Arguments

file_paths	character vector of paths to narrowPeak files. If named, those names will be used in output unless overriden by providing file_names.
file_names	character vector of names for output list. If not NULL will override any existing names for file_paths. Default is NULL.
extraCols	named character vector of classes. passed to rtracklayer::import for format = "BED". default is character().

Value

a named list of GRanges loaded from file_paths

Examples

```
bed_f = system.file("extdata/test_loading.bed",
    package = "seqsetvis", mustWork = TRUE)
easyLoad_bed(bed_f, "my_bed")
```

easyLoad_broadPeak *easyLoad_broadPeak takes a character vector of file paths to narrowPeak files from MACS2 and returns a named list of GRanges.*

Description

easyLoad_broadPeak takes a character vector of file paths to narrowPeak files from MACS2 and returns a named list of GRanges.

Usage

```
easyLoad_broadPeak(file_paths, file_names = NULL)
```

Arguments

file_paths	character vector of paths to narrowPeak files. If named, those names will be used in output unless overriden by providing file_names.
file_names	character vector of names for output list. If not NULL will override any existing names for file_paths. Default is NULL.

Value

a named list of GRanges loaded from file_paths

Examples

```
bp_f = system.file("extdata/test_loading.broadPeak",
    package = "seqsetvis", mustWork = TRUE)
easyLoad_broadPeak(bp_f, "my_broadPeak")
```

---

easyLoad_narrowPeak *easyLoad_narrowPeak takes a character vector of file paths to narrowPeak files from MACS2 and returns a named list of GRanges.*

Description

easyLoad_narrowPeak takes a character vector of file paths to narrowPeak files from MACS2 and returns a named list of GRanges.

Usage

```
easyLoad_narrowPeak(file_paths, file_names = NULL)
```

Arguments

file_paths	character vector of paths to narrowPeak files. If named, those names will be used in output unless overriden by providing file_names.
file_names	character vector of names for output list. If not NULL will override any existing names for file_paths. Default is NULL.

Value

a named list of GRanges loaded from file_paths

Examples

```
np_f = system.file("extdata/test_loading.narrowPeak",
    package = "seqsetvis", mustWork = TRUE)
easyLoad_narrowPeak(np_f, "my_narrowPeak")
```

`fetchBam` *fetch a bam file pileup with the ability to consider read extension to fragment size (fragLen)*

Description

fetch a bam file pileup with the ability to consider read extension to fragment size (fragLen)

Usage

```
fetchBam(bam_f, qgr, fragLen = NULL, target_strand = c("*", "+",
  "-")[1], max_dupes = Inf, splice_strategy = c("none", "ignore",
  "add", "only", "splice_count")[1], flip_strand = FALSE, ...)
```

Arguments

`bam_f`	character or BamFile to load
`qgr`	GRanges regions to fetchs
`fragLen`	numeric, NULL, or NA. if numeric, supplied value is used. if NULL, value is calculated with fragLen_calcStranded (default) if NA, raw bam pileup with no cross strand shift is returned.
`target_strand`	character. if one of "+" or "-", reads are filtered to match. ignored if any other value.
`max_dupes`	numeric >= 1. duplicate reads by strandd start position over this number are removed, Default is Inf.
`splice_strategy`	character, one of c("none", "ignore", "add", "only"). Default is "none" and split read alignments are asssumed not present. fragLen must be NA for any other value to be valid. "ignore" will not count spliced regions. "add" counts spliced regions along with others, "only" will only count spliced regions and ignore others.
`flip_strand`	if TRUE, strand alignment is flipped prior to fragLen extension. Default is FALSE.
`...`	passed to ScanBamParam(), can't be which or what.

Value

GRanges containing tag pileup values in score meta column. tags are optionally extended to fragment length (fragLen) prior to pile up.

`fragLen_calcStranded` *calculate fragLen from a bam file for specified regions*

Description

calculate fragLen from a bam file for specified regions

Usage

```
fragLen_calcStranded(bam_f, qgr, n_regions = 100,
  include_plot_in_output = FALSE, test_fragLen = seq(100, 400, 5),
  flip_strand = FALSE, ...)
```

Arguments

`bam_f`	character or BamFile. bam file to read from. .bai index file must be in same directory
`qgr`	GRanges. used as which for ScanBamParam. Can be NULL if it's REALLY important to load the entire bam, force_no_which = TRUE also required.
`n_regions`	numeric (integer) it's generally overkill to pull all regions at this stage and will slow calculation down. Default is 100.
`include_plot_in_output`	if TRUE ouptut is a list of fragLen and a ggplot showing values considered by calculation. Default is FALSE.
`test_fragLen`	numeric. The set of fragment lenghts to gather strand cross correlation for.
`flip_strand`	boolean. if TRUE strands that reads align to are swapped. This is typically only necessary if there was a mismatch between library chemistry and aligner settings. Default is FALSE.
`...`	passed to Rsamtools::ScanBamParam, can't be which or what.

Value

numeric fragment length

Examples

```
bam_file = system.file("extdata/test.bam",
    package = "seqsetvis")
qgr = CTCF_in_10a_overlaps_gr[1:5]
fragLen_calcStranded(bam_file, qgr)
#if plot is included, a list is returned, item 2 is the plot
fragLen_calcStranded(bam_file, qgr,
  include_plot_in_output = TRUE)[[2]]
```

fragLen_fromMacs2Xls *parse fragLen from MACS2 output*

Description

parse fragLen from MACS2 output

Usage

```
fragLen_fromMacs2Xls(macs2xls_file)
```

Arguments

macs2xls_file	character. an xls file output by MACS2 to parse frag length from

Value

numeric fragment length

Examples

```
xls_file = system.file("extdata/test_peaks.xls",
    package = "seqsetvis")
fragLen_fromMacs2Xls(xls_file)
```

getReadLength *determine the most common read length for input bam_file. uses 50 randomly selected regions from query_gr. If fewer than 20 reads are present, loads all of query_gr.*

Description

determine the most common read length for input bam_file. uses 50 randomly selected regions from query_gr. If fewer than 20 reads are present, loads all of query_gr.

Usage

```
getReadLength(bam_file, query_gr)
```

Arguments

bam_file	indexed bam file
query_gr	GRanges to read from bam file

Value

numeric of most common read length.

`ggellipse` *returns a ggplot with ellipses drawn using specified parameters used by ssvFeatureVenn and ssvFeatureEuler*

Description

uses eulerr's non-exported ellipse drawing coordinate function

Usage

```
ggellipse(xcentres, ycentres, r, r2 = r, phi = rep(0,
  length(xcentres)), circle_colors = NULL,
  group_names = LETTERS[seq_along(xcentres)], line_alpha = 1,
  fill_alpha = 0.3, line_width = 2, n_points = 200)
```

Arguments

`xcentres`	numeric x-coord of centers of ellipses
`ycentres`	numeric y-coord of centers of ellipses, must have same length as xcentres
`r`	numeric radius1 of ellipse, must have length of 1 or match length of xcentres
`r2`	numeric radius2 of ellipse, must have length of 1 or match length of xcentres. same as r by default.
`phi`	numeric phi of ellipse, must have length of 1 or match length of xcentres. 0 by default.
`circle_colors`	character of rcolors or hex colors or NULL. if null safeBrew of Dark2 is used
`group_names`	character/factor names of color/fill groups. capital letters by default.
`line_alpha`	numeric [0,1] alpha of lines, 1 by default
`fill_alpha`	numeric [0,1] alpha of fill, .3 by default.
`line_width`	numeric > 0. passed to size. 2 by default
`n_points`	integer > 1. number of points to approximate circle with. 200 by default

Value

a ggplot containing ellipses

Examples

```
ggellipse(xcentres = c(1, 1, 2),
    ycentres = c(2, 1, 1),
    r = c(1, 2, 1))
ggellipse(xcentres = c(1, 1, 2),
    ycentres = c(2, 1, 1),
    r = c(1, 2, 1),
    fill_alpha = 0,
    group_names = paste("set", 1:3))
ggellipse(xcentres = c(1, 1, 2),
    ycentres = c(2, 1, 1),
    r = c(1, 2, 1),
    circle_colors = c("red", "orange", "yellow"),
    line_alpha = 0,
    group_names = paste("set", 1:3))
```

harmonize_seqlengths *harmonize_seqlengths*

Description

ensures compatibility between seqlength of gr and bam_file based on header

Usage

```
harmonize_seqlengths(gr, bam_file)
```

Arguments

gr	GRanges, object to harmonize seqlengths for
bam_file	character, a path to a valid bam file

Value

gr with seqlengths matching bam_file

Examples

```
library(GenomicRanges)
gr = GRanges("chr1", IRanges(1, 100))
#seqlengths has not been set
seqlengths(gr)
bam = system.file("extdata/test.bam", package = "seqsetvis")
gr2 = harmonize_seqlengths(gr, bam)
#seqlengths now set
seqlengths(gr2)
```

prepare_fetch_GRanges *prepares GRanges for windowed fetching.*

Description

output GRanges parallels input with consistent width evenly divisible by win_size. Has warning if GRanges needed resizing, otherwise no warning and input GRanges is returned unchanged.

Usage

```
prepare_fetch_GRanges(qgr, win_size, min_quantile = 0.75,
  target_size = NULL)
```

Arguments

qgr	GRanges to prepare
win_size	numeric window size for fetch
min_quantile	numeric [0,1], lowest possible quantile value. Only relevant if target_size is not specified.
target_size	numeric final width of qgr if known. Default of NULL leads to quantile based determination of target_size.

Value

GRanges, either identical to qgr or with suitable consistent width applied.

Examples

```
qgr = prepare_fetch_GRanges(CTCF_in_10a_overlaps_gr, win_size = 50)
#no warning if qgr is already valid for windowed fetching
prepare_fetch_GRanges(qgr, win_size = 50)
```

---

quantileGRangesWidth *Quantile width determination strategy*

Description

Returns the lowest multiple of win_size greater than min_quantile quantile of width(qgr)

Usage

```
quantileGRangesWidth(qgr, min_quantile = 0.75, win_size = 1)
```

Arguments

qgr	GRanges to calculate quantile width for
min_quantile	numeric [0,1] the minimum quantile of width in qgr
win_size	numeric/integer >=1, returned value will be a multiple of this

Value

numeric that is >= min_quantile and evenly divisible by win_size

Examples

```
gr = CTCF_in_10a_overlaps_gr
quantileGRangesWidth(gr)
quantileGRangesWidth(gr, min_quantile = .5, win_size = 100)
```

---

safeBrew *allows RColorBrew to handle n values less than 3 and greater than 8 without warnings and return expected number of colors.*

Description

allows RColorBrew to handle n values less than 3 and greater than 8 without warnings and return expected number of colors.

Usage

```
safeBrew(n, pal = "Dark2")
```

Arguments

n	integer value of number of colors to make palette for
pal	palette recognized by RColorBrewer

Value

a character vector of hex coded colors o flength n from the color brewer palette pal

Examples

```
plot(1:2, rep(0, 2),  col = safeBrew(2, "dark2"), pch = 16, cex = 6)
plot(1:12, rep(0, 12),  col = safeBrew(12, "set1"), pch = 16, cex = 6)
plot(1:12, rep(0, 12),  col = safeBrew(12, "set2"), pch = 16, cex = 6)
plot(1:12, rep(0, 12),  col = safeBrew(12, "set3"), pch = 16, cex = 6)
```

---

set_list2memb *convert a list of sets, each list item should be a character vector denoting items in sets*

Description

convert a list of sets, each list item should be a character vector denoting items in sets

Usage

```
set_list2memb(set_list)
```

Arguments

set_list	a list of character vectors. default names will be added if missing

Value

converts list of characters/numeric to membership table matrix

---

shift_anchor *orients the relative position of x's zero value and extends ranges to be contiguous*

Description

orients the relative position of x's zero value and extends ranges to be contiguous

Usage

```
shift_anchor(score_dt, window_size, anchor)
```

Arguments

score_dt	data.table, GRanges() sufficient
window_size	numeric, window size used to generate socre_dt
anchor	character, one of c("center", "center_unstranded", "left", "left_unstranded")

Value

score_dt with x values shifted appropriately and start and end extended to make ranges contiguous

---

ssvFactorizeMembTable *Convert any object accepted by ssvMakeMembTable to a factor To avoid ambiguity,*

Description

see ssvMakeMembTable

Usage

```
ssvFactorizeMembTable(object)
```

Arguments

object	a valid object for conversion to a membership table and then factor

Value

a 2 column ("id" and "group") data.frame. "id" is factor of item names if any or simply order of items. "group" is a factor of set combinations

Examples

```
ssvFactorizeMembTable(CTCF_in_10a_overlaps_gr)
ssvFactorizeMembTable(list(1:4, 2:3, 4:6))
```

---

ssvFeatureBars *bar plots of set sizes*

Description

bar plots of set sizes

Usage

```
ssvFeatureBars(object, show_counts = TRUE, bar_colors = NULL,
  return_data = FALSE)
```

Arguments

object	passed to ssvMakeMembTable for conversion to membership table
show_counts	logical. should counts be displayed at the center of each bar. default is TRUE
bar_colors	character. rcolor or hex colors. default of NULL uses RColorBrewer Dark2.
return_data	logical. If TRUE, return value is no longer ggplot and is instead the data used to generate that plot. Default is FALSE.

Value

ggplot of bar plot of set sizes

Examples

```
ssvFeatureBars(list(1:3, 2:6))
ssvFeatureBars(CTCF_in_10a_overlaps_gr)
ssvFeatureBars(S4Vectors::mcols(CTCF_in_10a_overlaps_gr)[,2:3])
```

---

ssvFeatureBinaryHeatmap

binary heatmap indicating membership. heatmap is sorted by column left to right. change column order to reveal patterns

---

Description

binary heatmap indicating membership. heatmap is sorted by column left to right. change column order to reveal patterns

Usage

```
ssvFeatureBinaryHeatmap(object, raster_approximation = FALSE,
  true_color = "black", false_color = "#EFEFEF",
  raster_width_min = 1000, raster_height_min = 1000,
  return_data = FALSE)
```

Arguments

object	passed to ssvMakeMembTable
raster_approximation	If TRUE, instead of standard ggplot, write temporary raster png image and re-draw that as plot background. default is FALSE
true_color	character. rcolor or hex color used for TRUE values. default is "black".
false_color	character. rcolor or hex color used for TRUE values. default is "#EFEFEF", a gray.
raster_width_min	raster width will be minimum multiple of number of columns over this number. ignored if raster_approximation is FALSE.
raster_height_min	raster height will be minimum multiple of number of rows over this number ignored if raster_approximation is FALSE
return_data	logical. If TRUE, return value is no longer ggplot and is instead the data used to generate that plot. Default is FALSE.

Value

ggplot using geom_tile of membership table sorted from left to right.

Examples

```
ssvFeatureBinaryHeatmap(list(1:3, 2:6))
# horizontal version
ssvFeatureBinaryHeatmap(list(1:3, 2:6)) + coord_flip() +
  theme(axis.text.x = element_blank(), axis.text.y = element_text())
ssvFeatureBinaryHeatmap(CTCF_in_10a_overlaps_gr)
ssvFeatureBinaryHeatmap(S4Vectors::mcols(CTCF_in_10a_overlaps_gr)[,2:3])
ssvFeatureBinaryHeatmap(S4Vectors::mcols(CTCF_in_10a_overlaps_gr)[,3:2])
```

---

ssvFeatureEuler *Try to load a bed-like file and convert it to a GRanges object*

Description

Try to load a bed-like file and convert it to a GRanges object

Usage

```
ssvFeatureEuler(object, line_width = 2, shape = c("circle",
  "ellipse")[1], n_points = 200, fill_alpha = 0.3, line_alpha = 1,
  circle_colors = NULL, return_data = FALSE)
```

Arguments

object	A membership table
line_width	numeric, passed to size aesthetic to control line width
shape	shape argument passed to eulerr::euler
n_points	number of points to use for drawing ellipses, passed to eulerr:::ellipse
fill_alpha	numeric [0,1], alpha value for circle fill
line_alpha	numeric [0,1], alpha value for circle line
circle_colors	colors to choose from for circles. passed to ggplot2 color scales.
return_data	logical. If TRUE, return value is no longer ggplot and is instead the data used to generate that plot. Default is FALSE.

Value

ggplot of venneuler results

Examples

```
ssvFeatureEuler(list(1:3, 2:6))
ssvFeatureEuler(CTCF_in_10a_overlaps_gr)
ssvFeatureEuler(S4Vectors::mcols(CTCF_in_10a_overlaps_gr)[,2:3])
```

`ssvFeaturePie` *pie plot of set sizes*

Description

pie plot of set sizes

Usage

```
ssvFeaturePie(object, slice_colors = NULL, return_data = FALSE)
```

Arguments

`object`	object that ssvMakeMembTable can convert to logical matrix membership
`slice_colors`	colors to use for pie slices
`return_data`	logical. If TRUE, return value is no longer ggplot and is instead the data used to generate that plot. Default is FALSE.

Value

ggplot pie graph of set sizes

Examples

```
ssvFeaturePie(list(1:3, 2:6))
ssvFeaturePie(CTCF_in_10a_overlaps_gr)
ssvFeaturePie(S4Vectors::mcols(CTCF_in_10a_overlaps_gr)[,2:3])
```

`ssvFeatureVenn` *ggplot implementation of vennDiagram from limma package. currently limited at 3 sets*

Description

ggplot implementation of vennDiagram from limma package. currently limited at 3 sets

Usage

```
ssvFeatureVenn(object, group_names = NULL, counts_txt_size = 5,
  counts_as_labels = FALSE, show_outside_count = FALSE,
  line_width = 3, circle_colors = NULL, fill_alpha = 0.3,
  line_alpha = 1, counts_color = NULL, n_points = 200,
  return_data = FALSE)
```

Arguments

object	will be passed to ssvMakeMembTable for conversion to membership matrix
group_names	useful if names weren't provided or were lost in creating membership matrix
counts_txt_size	font size for count numbers
counts_as_labels	if TRUE, geom_label is used instead of geom_text. can be easier to read.
show_outside_count	if TRUE, items outside of all sets are counted outside. can be confusing.
line_width	uses size aesthetic to control line width of circles.
circle_colors	colors to use for circle line colors. Uses Dark2 set from RColorBrewer by default.
fill_alpha	alpha value to use for fill, defaults to .3.
line_alpha	numeric [0,1], alpha value for circle line
counts_color	character. single color to use for displaying counts
n_points	integer. number of points to approximate circle with. default is 200.
return_data	logical. If TRUE, return value is no longer ggplot and is instead the data used to generate that plot. Default is FALSE.

Value

ggplot venn diagram

Examples

```
ssvFeatureVenn(list(1:3, 2:6))
ssvFeatureVenn(CTCF_in_10a_overlaps_gr)
ssvFeatureVenn(S4Vectors::mcols(CTCF_in_10a_overlaps_gr)[,2:3])
```

---

ssvFetchBam

Iterates a character vector (ideally named) and calls ssvFetchBam.single *on each. Appends grouping variable to each resulting data.table and uses rbindlist to efficiently combine results*

Description

ssvFetchBam iteratively calls fetchWindowedBam.single. See ssvFetchBam.single for more info.

Usage

```
ssvFetchBam(file_paths, qgr, unique_names = NULL, win_size = 50,
  win_method = c("sample", "summary")[1],
  summary_FUN = stats::weighted.mean, fragLens = "auto",
  target_strand = c("*", "+", "-", "both")[1], flip_strand = FALSE,
  anchor = c("left", "left_unstranded", "center",
  "center_unstranded")[3], names_variable = "sample",
  return_data.table = FALSE, max_dupes = Inf,
  splice_strategy = c("none", "ignore", "add", "only",
  "splice_count")[1], n_cores = getOption("mc.cores", 1), ...)
```

Arguments

file_paths	character vector of file_paths to load from. Alternatively, file_paths can be a data.frame or data.table whose first column is a character vector of paths and additial columns will be used as metadata.
qgr	Set of GRanges to query. For valid results the width of each interval should be identical and evenly divisible by win_size.
unique_names	names to use in final data.table to designate source bigwig. Default is 'sample'
win_size	The window size that evenly divides widths in qgr.
win_method	character. one of c("sample", "summary"). Determines if viewGRangesWinSample_dt or viewGRangesWinSummary_dt is used to represent each region in qgr.
summary_FUN	function. only relevant if win_method is "summary". passed to viewGRangesWinSummary_dt.
fragLens	numeric. The fragment length to use to extend reads. The default value "auto" causes an automatic calculation from 100 regions in qgr. NA causes no extension of reads to fragment size.
target_strand	character. One of c("*", "+", "-"). Controls filtering of reads by strand. Default of "*" combines both strands.
flip_strand	boolean. if TRUE strands are flipped.
anchor	character, one of c("center", "center_unstranded", "left", "left_unstranded")
names_variable	The column name where unique_names are stored.
return_data.table	logical. If TRUE the internal data.table is returned instead of GRanges. Default is FALSE.
max_dupes	numeric >= 1. duplicate reads by strandd start position over this number are removed, Default is Inf.
splice_strategy	character, one of c("none", "ignore", "add", "only", "splice_count"). Default is "none" and spliced alignment are asssumed not present. fragLen must be NA for any other value to be valid. "ignore" will not count spliced regions. add" counts spliced regions along with others, "only" will only count spliced regions and ignore others.
n_cores	integer number of cores to use.
...	passed to Rsamtools::ScanBamParam() Uses mc.cores option if not supplied.

Details

if qgr contains the range chr1:1-100 and win_size is 10, values from positions chr1 5,15,25...85, and 95 will be retrieved from bw_file

Value

A tidy formatted GRanges (or data.table if specified) containing fetched values.

Examples

```
if(Sys.info()['sysname'] != "Windows"){
library(GenomicRanges)
bam_f = system.file("extdata/test.bam",
    package = "seqsetvis", mustWork = TRUE)
bam_files = c("a" = bam_f, "b" = bam_f)
```

```
qgr = CTCF_in_10a_overlaps_gr[1:5]
bw_gr = ssvFetchBam(bam_files, qgr, win_size = 10)
bw_gr2 = ssvFetchBam(as.list(bam_files), qgr, win_size = 10)

bw_dt = ssvFetchBam(bam_files, qgr, win_size = 10,
    return_data.table = TRUE)
}
```

`ssvFetchBam.single` *fetch a windowed version of a bam file, returns GRanges*

Description

fetch a windowed version of a bam file, returns GRanges

Usage

```
ssvFetchBam.single(bam_f, qgr, win_size = 50, win_method = c("sample",
  "summary")[1], summary_FUN = stats::weighted.mean, fragLen = NULL,
  target_strand = c("*", "+", "-", "both")[1], anchor = c("left",
  "left_unstranded", "center", "center_unstranded")[3],
  return_data.table = FALSE, max_dupes = Inf,
  splice_strategy = c("none", "ignore", "add", "only",
  "splice_count")[1], flip_strand = FALSE, ...)
```

Arguments

`bam_f`	character or BamFile to load
`qgr`	GRanges regions to fetchs
`win_size`	numeric >=1. pileup grabbed every win_size bp for win_method sample. If win_method is summary, this is the number of windows used (confusing, sorry).
`win_method`	character. one of c("sample", "summary"). Determines if `viewGRangesWinSample_dt` or `viewGRangesWinSummary_dt` is used to represent each region in qgr.
`summary_FUN`	function. only relevant if win_method is "summary". passed to `viewGRangesWinSummary_dt`.
`fragLen`	numeric, NULL, or NA. if numeric, supplied value is used. if NULL, value is calculated with fragLen_calcStranded if NA, raw bam pileup with no cross strand shift is returned.
`target_strand`	character. if one of "+" or "-", reads are filtered accordingly. ignored if any other value.
`anchor`	character, one of c("center", "center_unstranded", "left", "left_unstranded")
`return_data.table`	logical. If TRUE the internal data.table is returned instead of GRanges. Default is FALSE.
`max_dupes`	numeric >= 1. duplicate reads by strandd start position over this number are removed, Default is Inf.
`splice_strategy`	character, one of c("none", "ignore", "add", "only", "splice_count"). Default is "none" and spliced alignment are asssumed not present. fragLen must be NA for any other value to be valid. "ignore" will not count spliced regions. add" counts spliced regions along with others, "only" will only count spliced regions and ignore others.

flip_strand	if TRUE, strand alignment is flipped prior to fragLen extension. Default is FALSE.
...	passed to Rsamtools::ScanBamParam()

Value

tidy GRanges (or data.table if specified) with pileups from bam file. pileup is calculated only every win_size bp.

---

ssvFetchBamPE *ssvFetchBam for paired-end ChIP-seq files.*

Description

Iterates a character vector (ideally named) and calls ssvFetchBamPE.single on each. Appends grouping variable to each resulting data.table and uses rbindlist to efficiently combine results

Usage

```
ssvFetchBamPE(file_paths, qgr, unique_names = NULL, win_size = 50,
  win_method = c("sample", "summary")[1],
  summary_FUN = stats::weighted.mean, anchor = c("left",
  "left_unstranded", "center", "center_unstranded")[3],
  names_variable = "sample", return_data.table = FALSE,
  max_dupes = Inf, n_cores = getOption("mc.cores", 1), min_isize = 1,
  max_isize = Inf, return_unprocessed = FALSE, ...)
```

Arguments

file_paths	character vector of file_paths to load from. Alternatively, file_paths can be a data.frame or data.table whose first column is a character vector of paths and additial columns will be used as metadata.
qgr	Set of GRanges to query. For valid results the width of each interval should be identical and evenly divisible by win_size.
unique_names	names to use in final data.table to designate source bigwig. Default is 'sample'
win_size	The window size that evenly divides widths in qgr.
win_method	character. one of c("sample", "summary"). Determines if viewGRangesWinSample_dt or viewGRangesWinSummary_dt is used to represent each region in qgr.
summary_FUN	function. only relevant if win_method is "summary". passed to viewGRangesWinSummary_dt.
anchor	character, one of c("center", "center_unstranded", "left", "left_unstranded")
names_variable	The column name where unique_names are stored.
return_data.table	logical. If TRUE the internal data.table is returned instead of GRanges. Default is FALSE.
max_dupes	numeric >= 1. duplicate reads by strandd start position over this number are removed, Default is Inf.
n_cores	integer number of cores to use.

min_isize	integer. Read pairs must have an isize greater than or equal to this value. Default is 1.
max_isize	integer. Read pairs must have an isize less than or equal to this value. Default is Inf.
return_unprocessed	boolean. if TRUE returns read alignment in data.table. Default is FALSE.
...	passed to Rsamtools::ScanBamParam() Uses mc.cores option if not supplied.

Details

#' In contrast to ssvFetchBam, extension of reads to estimated fragment size is not an issue as each read pair represents a fragment of exact size.

ssvFetchBamPE iteratively calls fetchWindowedBam.single. See ssvFetchBamPE.single for more info.

if qgr contains the range chr1:1-100 and win_size is 10, values from positions chr1 5,15,25...85, and 95 will be retrieved from bw_file

Value

A tidy formatted GRanges (or data.table if specified) containing fetched values.

Examples

```
if(Sys.info()['sysname'] != "Windows"){
library(GenomicRanges)
bam_f = system.file("extdata/Bcell_PE.mm10.bam",
    package = "seqsetvis", mustWork = TRUE)
bam_files = c("a" = bam_f, "b" = bam_f)
data("Bcell_peaks")
qgr = Bcell_peaks
bw_gr = ssvFetchBamPE(bam_files, qgr, win_size = 10)
bw_gr2 = ssvFetchBamPE(as.list(bam_files), qgr, win_size = 10)

bw_dt = ssvFetchBamPE(bam_files, qgr, win_size = 10,
    return_data.table = TRUE)
}
```

---

ssvFetchBamPE.single *fetch a windowed version of a paired-end bam file, returns GRanges In contrast to ssvFetchBam, extension of reads to estimated fragment size is not an issue as each read pair represents a fragment of exact size.*

Description

fetch a windowed version of a paired-end bam file, returns GRanges In contrast to ssvFetchBam, extension of reads to estimated fragment size is not an issue as each read pair represents a fragment of exact size.

Usage

```
ssvFetchBamPE.single(bam_f, qgr, win_size = 50,
  win_method = c("sample", "summary")[1],
  summary_FUN = stats::weighted.mean, anchor = c("left",
  "left_unstranded", "center", "center_unstranded")[3],
  return_data.table = FALSE, max_dupes = Inf, min_isize = 1,
  max_isize = Inf, return_unprocessed = FALSE, ...)
```

Arguments

bam_f	character or BamFile to load
qgr	GRanges regions to fetchs
win_size	numeric >=1. pileup grabbed every win_size bp for win_method sample. If win_method is summary, this is the number of windows used (confusing, sorry).
win_method	character. one of c("sample", "summary"). Determines if viewGRangesWinSample_dt or viewGRangesWinSummary_dt is used to represent each region in qgr.
summary_FUN	function. only relevant if win_method is "summary". passed to viewGRangesWinSummary_dt.
anchor	character, one of c("center", "center_unstranded", "left", "left_unstranded")
return_data.table	logical. If TRUE the internal data.table is returned instead of GRanges. Default is FALSE.
max_dupes	numeric >= 1. duplicate reads by strandd start position over this number are removed, Default is Inf.
min_isize	integer. Read pairs must have an isize greater than or equal to this value. Default is 1.
max_isize	integer. Read pairs must have an isize less than or equal to this value. Default is Inf.
return_unprocessed	boolean. if TRUE returns read alignment in data.table. Default is FALSE.
...	passed to Rsamtools::ScanBamParam()

Value

tidy GRanges (or data.table if specified) with pileups from bam file. pileup is calculated only every win_size bp.

---

ssvFetchBigwig *Iterates a character vector (ideally named) and calls* ssvFetchBigwig.single *on each. Appends grouping variable to each resulting data.table and uses rbindlist to efficiently combine results.*

Description

ssvFetchBigwig iteratively calls fetchWindowedBigwig.single. See ssvFetchBigwig.single for more info.

Usage

```
ssvFetchBigwig(file_paths, qgr, unique_names = NULL,
  names_variable = "sample", win_size = 50, win_method = c("sample",
  "summary")[1], summary_FUN = stats::weighted.mean, anchor = c("left",
  "left_unstranded", "center", "center_unstranded")[3],
  return_data.table = FALSE, n_cores = getOption("mc.cores", 1))
```

Arguments

file_paths	character vector of file_paths to load from. Alternatively, file_paths can be a data.frame or data.table whose first column is a character vector of paths and additial columns will be used as metadata.
qgr	Set of GRanges to query. For valid results the width of each interval should be identical and evenly divisible by win_size.
unique_names	names to use in final data.table to designate source bigwig.
names_variable	The column name where unique_names are stored. Default is 'sample'
win_size	The window size that evenly divides widths in qgr.
win_method	character. one of c("sample", "summary"). Determines if viewGRangesWinSample_dt or viewGRangesWinSummary_dt is used to represent each region in qgr.
summary_FUN	function. only relevant if win_method is "summary". passed to viewGRangesWinSummary_dt.
anchor	character, one of c("center", "center_unstranded", "left", "left_unstranded")
return_data.table	logical. If TRUE the internal data.table is returned instead of GRanges. Default is FALSE.
n_cores	integer number of cores to use. Uses mc.cores option if not supplied.

Details

if qgr contains the range chr1:1-100 and win_size is 10, values from positions chr1 5,15,25...85, and 95 will be retrieved from bw_file

Value

A tidy formatted GRanges (or data.table if specified) containing fetched values.

Examples

```
if(Sys.info()['sysname'] != "Windows"){
library(GenomicRanges)
bw_f = system.file("extdata/test_loading.bw",
    package = "seqsetvis", mustWork = TRUE)
bw_files = c("a" = bw_f, "b" = bw_f)
qgr = GRanges("chrTest", IRanges(1, 30))
bw_gr = ssvFetchBigwig(bw_files, qgr, win_size = 10)
bw_gr2 = ssvFetchBigwig(as.list(bw_files), qgr, win_size = 10)

bw_dt = ssvFetchBigwig(bw_files, qgr, win_size = 10,
    return_data.table = TRUE)
}
```

`ssvFetchBigwig.single` *Fetch values from a bigwig appropriate for heatmaps etc.*

Description

`ssvFetchBigwig.single` Gets values for each region of the query GRanges (`qgr`). Values correspond to the center of each window of size `win_size`. A tidy formatted data.table object is returned suitable for plotting using ggplots.

Usage

```
ssvFetchBigwig.single(bw_file, qgr, win_size = 50,
  win_method = c("sample", "summary")[1],
  summary_FUN = stats::weighted.mean, anchor = c("left",
  "left_unstranded", "center", "center_unstranded")[3],
  return_data.table = FALSE)
```

Arguments

`bw_file`	The character vector path to bigwig files to read from.
`qgr`	Set of GRanges to query. For valid results the width of each interval should be identical and evenly divisible by `win_size`.
`win_size`	The window size that evenly divides widths in `qgr`.
`win_method`	character. one of c("sample", "summary"). Determines if `viewGRangesWinSample_dt` or `viewGRangesWinSummary_dt` is used to represent each region in qgr.
`summary_FUN`	function. only relevant if win_method is "summary". passed to `viewGRangesWinSummary_dt`.
`anchor`	character, one of c("center", "center_unstranded", "left", "left_unstranded")
`return_data.table`	logical. If TRUE the internal data.table is returned instead of GRanges. Default is FALSE.

Details

if `qgr` contains the range chr1:1-100 and `win_size` is 10, values from positions chr1 5,15,25...85, and 95 will be retrieved from `bw_file`

Value

A GRanges (or data.table if specified) containing fetched values.

ssvFetchGRanges *Fetch coverage values for a list of GRanges.*

Description

ssvFetchGRanges Gets coverage values for each region of the query GRanges (qgr). Values correspond to the center of each window of size win_size. A tidy formatted data.table object is returned suitable for plotting using ggplots.

Usage

```
ssvFetchGRanges(grs, qgr, file_attribs = data.frame(matrix(0, nrow =
  length(grs), ncol = 0)), unique_names = names(grs),
  names_variable = "sample", win_size = 50, win_method = c("sample",
  "summary")[1], summary_FUN = function(x, w) max(x),
  target_strand = c("*", "+", "-", "both")[1], use_coverage = NULL,
  attrib_var = "score", fill_value = 0, anchor = c("left",
  "left_unstranded", "center", "center_unstranded")[3],
  return_data.table = FALSE, n_cores = getOption("mc.cores", 1))
```

Arguments

grs	a list of GRanges for which to calculate coverage.
qgr	Set of GRanges to query. For valid results the width of each interval should be identical and evenly divisible by win_size.
file_attribs	data.frame (1 row per item in grs) containing attributes to append to results.
unique_names	The column name where unique_names are stored. Default is 'sample'
names_variable	The column name where unique_names are stored. Default is 'sample'
win_size	The window size that evenly divides widths in qgr.
win_method	character. one of c("sample", "summary"). Determines if viewGRangesWinSample_dt or viewGRangesWinSummary_dt is used to represent each region in qgr.
summary_FUN	function. only relevant if win_method is "summary". passed to viewGRangesWinSummary_dt.
target_strand	character. if one of "+" or "-", reads are filtered to match. ignored if any other value.
use_coverage	boolean or NULL, if TRUE, query regions are scored by the number of intervals overlapping. Default of NULL checks if attrib_var is "score" and uses coverage if so.
attrib_var	character, column in mcols of GRanges to pull values from. Default of "score" is compatible with internal coverage calculation or bedgraph-like files.
fill_value	numeric or character value to use where queried regions are empty. Default is 0 and appropriate for both calculated coverage and bedgraph/bigwig like files. Will automatically switch to "MISSING" if data is guessed to be qualitative.
anchor	character, one of c("center", "center_unstranded", "left", "left_unstranded")
return_data.table	logical. If TRUE the internal data.table is returned instead of GRanges. Default is FALSE.
n_cores	integer number of cores to use. Uses mc.cores option if not supplied.

Value

A tidy formatted GRanges (or data.table if specified) containing fetched values.

Examples

```
ssvFetchGRanges(CTCF_in_10a_narrowPeak_grs, CTCF_in_10a_overlaps_gr, win_size = 200)
```

`ssvFetchSignal` *signal loading framework*

Description

Does nothing unless load_signal is overridden to carry out reading data from file_paths (likely via the appropriate ssvFetch* function, ie. `ssvFetchBigwig` or `ssvFetchBam`

Usage

```
ssvFetchSignal(file_paths, qgr, unique_names = NULL,
  names_variable = "sample", win_size = 50, win_method = c("sample",
  "summary")[1], return_data.table = FALSE, load_signal = function(f,
  nam, qgr) {     warning("nothing happened, ", "supply a function to",
  "load_signal parameter.") }, n_cores = getOption("mc.cores", 1))
```

Arguments

`file_paths`	character vector of file_paths to load from. Alternatively, file_paths can be a data.frame or data.table whose first column is a character vector of paths and additial columns will be used as metadata.
`qgr`	GRanges of intervals to return from each file
`unique_names`	unique file ids for each file in file_paths. Default is names of file_paths vector
`names_variable`	character, variable name for column containing unique_names entries. Default is "sample"
`win_size`	numeric/integer window size resolution to load signal at. Default is 50.
`win_method`	character. one of c("sample", "summary"). Determines if `viewGRangesWinSample_dt` or `viewGRangesWinSummary_dt` is used to represent each region in qgr.
`return_data.table`	logical. If TRUE data.table is returned instead of GRanges, the default.
`load_signal`	function taking f, nam, and qgr arguments. f is from file_paths, nam is from unique_names, and qgr is qgr. See details.
`n_cores`	integer number of cores to use. Uses mc.cores option if not supplied.

Details

load_signal is passed f, nam, and qgr and is executed in the environment where load_signal is defined. See `ssvFetchBigwig` and `ssvFetchBam` for examples.

Value

A GRanges with values read from file_paths at intervals of win_size. Originating file is coded by unique_names and assigned to column of name names_variable. Output is data.table is return_data.table is TRUE.

Examples

```
library(GenomicRanges)
bam_f = system.file("extdata/test.bam",
    package = "seqsetvis", mustWork = TRUE)
bam_files = c("a" = bam_f, "b" = bam_f)
qgr = CTCF_in_10a_overlaps_gr[1:2]
qgr = resize(qgr, 500, "center")
load_bam = function(f, nam, qgr) {
    message("loading ", f, " ...")
    dt = seqsetvis:::ssvFetchBam.single(bam_f = f,
                      qgr = qgr,
                      win_size = 50,
                      fragLen = NULL,
                      target_strand = "*",
                      return_data.table = TRUE)
    dt[["sample"]] = nam
    message("finished loading ", nam, ".")
    dt
}
ssvFetchSignal(bam_files, qgr, load_signal = load_bam)
```

`ssvMakeMembTable` *generic for methods to convert various objects to a logical matrix indicating membership of items (rows) in sets (columns)*

Description

generic for methods to convert various objects to a logical matrix indicating membership of items (rows) in sets (columns)

list of character vectors input

GRangesList input

GRanges with mcols input

DataFrame input

matrix of logicals, membership table

data.frame input, final output The final method for all inputs, checks column names and returns logical matrix

Usage

```
ssvMakeMembTable(object)

## S4 method for signature 'list'
ssvMakeMembTable(object)
```

```
## S4 method for signature 'GRangesList'
ssvMakeMembTable(object)

## S4 method for signature 'GRanges'
ssvMakeMembTable(object)

## S4 method for signature 'DataFrame'
ssvMakeMembTable(object)

## S4 method for signature 'matrix'
ssvMakeMembTable(object)

## S4 method for signature 'data.frame'
ssvMakeMembTable(object)
```

Arguments

object	the object to convert. Supported types: list (of character or GRanges), GRanges with membership table metadata, GRangesList, data.frame/matrix/DataFrame of membership table

Value

a logical matrix indicating membership of items (rows) in sets (columns)

Examples

```
char_list = list(letters[1:3], letters[2:4])
ssvMakeMembTable(char_list)
library(GenomicRanges)
gr_list = list(GRanges("chr1", IRanges(1:3*2, 1:3*2)),
    GRanges("chr1", IRanges(2:4*2, 2:4*2)))
ssvMakeMembTable(gr_list)
library(GenomicRanges)
gr_list = list(GRanges("chr1", IRanges(1:3*2, 1:3*2)),
    GRanges("chr1", IRanges(2:4*2, 2:4*2)))
ssvMakeMembTable(GRangesList(gr_list))
gr = GRanges("chr1", IRanges(1:3*2, 1:3*2))
gr$set_a = c(TRUE, TRUE, FALSE)
gr$set_b = c(FALSE, TRUE, TRUE)
ssvMakeMembTable(gr)
gr = GRanges("chr1", IRanges(1:3*2, 1:3*2))
gr$set_a = c(TRUE, TRUE, FALSE)
gr$set_b = c(FALSE, TRUE, TRUE)
ssvMakeMembTable(mcols(gr))
memb_mat =  matrix(c(TRUE, TRUE, FALSE, FALSE, TRUE, FALSE, TRUE, FALSE),
    ncol = 2, byrow = FALSE)
ssvMakeMembTable(memb_mat)
memb_df = data.frame(a = c(TRUE, TRUE, FALSE, FALSE),
    b = c(TRUE, FALSE, TRUE, FALSE))
ssvMakeMembTable(memb_df)
```

ssvOverlapIntervalSets

Intersect a list of GRanges to create a single GRanges object of merged ranges including metadata describing overlaps per input GRanges

Description

Intersect a list of GRanges to create a single GRanges object of merged ranges including metadata describing overlaps per input GRanges

Usage

```
ssvOverlapIntervalSets(grs, ext = 0, use_first = FALSE)
```

Arguments

grs	A list of GRanges
ext	An integer specifying how far to extend ranges before merging. in effect, ranges withing 2*ext of one another will be joined during the merge
use_first	A logical. If True, instead of merging all grs, only use first and add metadata logicals for others.

Value

GRanges with metadata columns describing overlap of input grs

Examples

```
library(GenomicRanges)
a = GRanges("chr1", IRanges(1:7*10, 1:7*10))
b = GRanges("chr1", IRanges(5:10*10, 5:10*10))
ssvOverlapIntervalSets(list(a, b))
```

ssvSignalBandedQuantiles

plot profiles from bigwigs

Description

plot profiles from bigwigs

Usage

```
ssvSignalBandedQuantiles(bw_data, y_ = "y", x_ = "x", by_ = "fake",
  hsv_reverse = FALSE, hsv_saturation = 1, hsv_value = 1,
  hsv_grayscale = FALSE, hsv_hue_min = 0, hsv_hue_max = 0.7,
  hsv_symmetric = FALSE, n_quantile = 18, quantile_min = 0.05,
  quantile_max = 0.95, return_data = FALSE)
```

Arguments

bw_data	a GRanges or data.table of bigwig signal. As returned from ssvFetchBam and ssvFetchBigwig
y_	the variable name in bw_data for y axis in plot
x_	the variable name in bw_data for x axis in plot
by_	the variable name in bw_data to facet on
hsv_reverse	logical, should color scale be reversed? default FALSE
hsv_saturation	numeric [0, 1] saturation for color scale. default 1
hsv_value	numeric [0, 1] value for color scale. default 1
hsv_grayscale	logical, if TRUE gray() is used instead of rainbow(). default FALSE
hsv_hue_min	numeric [0, hsv_hue_max) hue min of color scale
hsv_hue_max	numeric (hsv_hue_min, 1] hue max of color scale
hsv_symmetric	if TRUE, colorscale is symmetrical, default FALSE.
n_quantile	number of evenly size quantile bins
quantile_min	the lowest quantile start
quantile_max	the highest quantile end
return_data	logical. If TRUE, return value is no longer ggplot and is instead the data used to generate that plot. Default is FALSE.

Value

ggplot object using ribbon plots to show quantile distributions

Examples

```
#rainbow colors
qgr = CTCF_in_10a_profiles_gr
ssvSignalBandedQuantiles(qgr)
#grayscale
ssvSignalBandedQuantiles(qgr, hsv_grayscale = TRUE,
    hsv_symmetric = TRUE, hsv_reverse = TRUE)
#using "by_" per sample
ssvSignalBandedQuantiles(qgr, hsv_grayscale = TRUE,
    hsv_symmetric = TRUE, hsv_reverse = TRUE, by_ = "sample")
#adding spline smoothing
splined = applySpline(qgr, n = 10,
    by_ = c("id", "sample"))
ssvSignalBandedQuantiles(splined, n_quantile = 50,
    quantile_min = .25, quantile_max = .75,
    hsv_symmetric = TRUE, hsv_reverse = TRUE, by_ = "sample")
```

`ssvSignalClustering` *clustering as for a heatmap*

Description

clustering as for a heatmap

Usage

```
ssvSignalClustering(bw_data, nclust = 6, row_ = "id", column_ = "x",
  fill_ = "y", facet_ = "sample", cluster_ = "cluster_id",
  max_rows = 500, max_cols = 100, clustering_col_min = -Inf,
  clustering_col_max = Inf, dcast_fill = NA)
```

Arguments

`bw_data`	a GRanges or data.table of bigwig signal. As returned from `ssvFetchBam` and `ssvFetchBigwig`
`nclust`	number of clusters
`row_`	variable name mapped to row, likely peak id or gene name for ngs data
`column_`	varaible mapped to column, likely bp position for ngs data
`fill_`	numeric variable to map to fill
`facet_`	variable name to facet horizontally by
`cluster_`	variable name to use for cluster info
`max_rows`	for speed rows are sampled to 500 by default, use Inf to plot full data
`max_cols`	for speed columns are sampled to 100 by default, use Inf to plot full data
`clustering_col_min`	numeric minimum for col range considered when clustering, default in -Inf
`clustering_col_max`	numeric maximum for col range considered when clustering, default in Inf
`dcast_fill`	value to supply to dcast fill argument. default is NA.

Value

data.table of signal profiles, ready for ssvSignalHeatmap

Examples

```
clust_dt = ssvSignalClustering(CTCF_in_10a_profiles_gr)
ssvSignalHeatmap(clust_dt)

clust_dt2 = ssvSignalClustering(CTCF_in_10a_profiles_gr, nclust = 2)
ssvSignalHeatmap(clust_dt2)

#clustering can be targetted to a specific part of the region
clust_dt3 = ssvSignalClustering(CTCF_in_10a_profiles_gr, nclust = 2,
    clustering_col_min = -250, clustering_col_max = -150)
ssvSignalHeatmap(clust_dt3)
clust_dt4 = ssvSignalClustering(CTCF_in_10a_profiles_gr, nclust = 2,
    clustering_col_min = 150, clustering_col_max = 250)
ssvSignalHeatmap(clust_dt4)
```

`ssvSignalHeatmap` *heatmap style representation of membership table. instead of clustering, each column is sorted starting from the left.*

Description

heatmap style representation of membership table. instead of clustering, each column is sorted starting from the left.

Usage

```
ssvSignalHeatmap(bw_data, nclust = 6, perform_clustering = c("auto",
  "yes", "no")[1], row_ = "id", column_ = "x", fill_ = "y",
  facet_ = "sample", cluster_ = "cluster_id", max_rows = 500,
  max_cols = 100, clustering_col_min = -Inf,
  clustering_col_max = Inf, return_data = FALSE)
```

Arguments

`bw_data`	a GRanges or data.table of bigwig signal. As returned from `ssvFetchBam` and `ssvFetchBigwig`
`nclust`	number of clusters
`perform_clustering`	should clustering be done? default is auto. auto considers if row_ has been ordered by being a factor and if cluster_ is a numeric.
`row_`	variable name mapped to row, likely peak id or gene name for ngs data
`column_`	varaible mapped to column, likely bp position for ngs data
`fill_`	numeric variable to map to fill
`facet_`	variable name to facet horizontally by
`cluster_`	variable name to use for cluster info
`max_rows`	for speed rows are sampled to 500 by default, use Inf to plot full data
`max_cols`	for speed columns are sampled to 100 by default, use Inf to plot full data
`clustering_col_min`	numeric minimum for col range considered when clustering, default in -Inf
`clustering_col_max`	numeric maximum for col range considered when clustering, default in Inf
`return_data`	logical. If TRUE, return value is no longer ggplot and is instead the data used to generate that plot. Default is FALSE.

Value

ggplot heatmap of signal profiles, facetted by sample

Examples

```
#the simplest use
ssvSignalHeatmap(CTCF_in_10a_profiles_gr)

#clustering can be done manually beforehand
clust_dt = ssvSignalClustering(CTCF_in_10a_profiles_gr, nclust = 3)
ssvSignalHeatmap(clust_dt)
```

`ssvSignalLineplot` *construct line type plots where each region in each sample is represented*

Description

construct line type plots where each region in each sample is represented

Usage

```
ssvSignalLineplot(bw_data, x_ = "x", y_ = "y", color_ = "sample",
  sample_ = "sample", region_ = "id", group_ = "auto_grp",
  line_alpha = 1, facet_ = "auto_facet", facet_method = facet_wrap,
  spline_n = NULL, return_data = FALSE)
```

Arguments

`bw_data`	a GRanges or data.table of bigwig signal. As returned from `ssvFetchBam` and `ssvFetchBigwig`
`x_`	variable name mapped to x aesthetic, x by default.
`y_`	variable name mapped to y aesthetic, y by default.
`color_`	variable name mapped to color aesthetic, sample by default.
`sample_`	variable name, along with region_ used to group and facet by default, change group_ or facet_ to override.
`region_`	variable name, along with sample_ used to group and facet by default, change group_ or facet_ to override.
`group_`	group aesthetic keeps lines of geom_path from mis-connecting. auto_grp by default which combines sample_ and region_. probably shouldn't change.
`line_alpha`	alpha value for lines. default is 1.
`facet_`	facetting divides up plots. auto_facet by default which combines sample_ and region_. if overriding facet_method with facet_grid, make sure to include ~ between two variables, ie. "a~b", ".~b", "a~."
`facet_method`	ggplot2 facetting method or wrapper for same, facet_wrap by default.
`spline_n`	if not NULL, applySpline will be called with n = spline_n. default is NULL.
`return_data`	logical. If TRUE, return value is no longer ggplot and is instead the data used to generate that plot. Default is FALSE.

Value

ggplot of signal potentially facetted by region and sample

Examples

```
bw_gr = CTCF_in_10a_profiles_gr
ssvSignalLineplot(subset(bw_gr, bw_gr$id %in% seq_len(3)), facet_ = "sample")
ssvSignalLineplot(subset(bw_gr, bw_gr$id %in% seq_len(3)),
    facet_ = "sample~.",
    facet_method = facet_grid)
ssvSignalLineplot(subset(bw_gr, bw_gr$id %in% seq_len(3)),
```

```
    facet_ = paste("sample", "~", "id"), facet_method = facet_grid)
ssvSignalLineplot(subset(bw_gr, bw_gr$id %in% seq_len(3)))
ssvSignalLineplot(subset(bw_gr, bw_gr$id %in% seq_len(3)), facet_ = "id")
ssvSignalLineplot(subset(bw_gr, bw_gr$id %in% seq_len(3)),
    facet_ = "id", spline_n = 10)
```

`ssvSignalLineplotAgg` *aggregate line signals in a single line plot*

Description

aggregate line signals in a single line plot

Usage

```
ssvSignalLineplotAgg(bw_data, x_ = "x", y_ = "y", sample_ = "sample",
  color_ = sample_, group_ = sample_, agg_fun = mean,
  spline_n = NULL, return_data = FALSE)
```

Arguments

bw_data	a GRanges or data.table of bigwig signal. As returned from ssvFetchBam and ssvFetchBigwig
x_	variable name mapped to x aesthetic, x by default.
y_	variable name mapped to y aesthetic, y by default.
sample_	variable name, along with region_ used to group by default,
color_	variable name mapped to color aesthetic, sample_ by default. change group_ to override.
group_	group aesthetic keeps lines of geom_path from mis-connecting. Most useful if you need to supply a variable to later facet upon. Defaults to value of sample_.
agg_fun	the aggregation function to apply by sample_ and x_, default is mean
spline_n	if not NULL, applySpline will be called with n = spline_n. default is NULL.
return_data	logical. If TRUE, return value is no longer ggplot and is instead the data used to generate that plot. Default is FALSE.

Value

ggplot of signal aggregated with agg_fun() by sample.

Examples

```
bw_gr = CTCF_in_10a_profiles_gr
ssvSignalLineplotAgg(bw_gr) +
    labs(title = "agg regions by sample.")
ssvSignalLineplotAgg(CTCF_in_10a_profiles_gr, spline_n = 10) +
    labs(title = "agg regions by sample, with spline smoothing.")
ssvSignalLineplotAgg(subset(bw_gr, bw_gr$id %in% seq_len(10)),
    sample_ = "id", color_ = "id") +
    labs(title = "agg samples by region id (weird)")
ssvSignalLineplotAgg(subset(bw_gr, bw_gr$id %in% seq_len(10)), sample_ = "id",
    color_ = "id", spline_n = 10) +
    labs(title = "agg samples by region id (weird), with spline smoothing")
```

`ssvSignalScatterplot` *maps signal from 2 sample profiles to the x and y axis. axes are standard or "volcano" min XY vs fold-change Y/X*

Description

maps signal from 2 sample profiles to the x and y axis. axes are standard or "volcano" min XY vs fold-change Y/X

Usage

```
ssvSignalScatterplot(bw_data, x_name, y_name, color_table = NULL,
  value_variable = "y", xy_variable = "sample", value_function = max,
  by_ = "id", plot_type = c("standard", "volcano")[1],
  show_help = FALSE, fixed_coords = TRUE, return_data = FALSE)
```

Arguments

`bw_data`	a GRanges or data.table of bigwig signal. As returned from `ssvFetchBam` and `ssvFetchBigwig`
`x_name`	sample name to map to x-axis, must be stored in variable specified in `xy_variable`
`y_name`	sample name to map to y-axis, must be stored in variable specified in `xy_variable`
`color_table`	data.frame with 2 columns, one of which must be named "group" and gets mapped to color. The other column must be the same as by_ parameter and is used for merging.
`value_variable`	variable name that stores numeric values for plotting, default is "y"
`xy_variable`	variable name that stores sample, must contain entires for `x_name` and `y_name`
`value_function`	a function to apply to `value_variable` in all combintations of `by_` per `x_name` and `y_name`
`by_`	variables that store individual measurement ids
`plot_type`	standard or volcano, default is "standard"
`show_help`	if TRUE overlay labels to aid plot interpretation, default is FALSE
`fixed_coords`	if TRUE coordinate system is 1:1 ratio, default is TRUE
`return_data`	logical. If TRUE, return value is no longer ggplot and is instead the data used to generate that plot. Default is FALSE.

Value

ggplot of points comparing signal from 2 samples

Examples

```
ssvSignalScatterplot(CTCF_in_10a_profiles_gr,
    x_name = "MCF10A_CTCF", y_name = "MCF10AT1_CTCF")
ssvSignalScatterplot(CTCF_in_10a_profiles_gr,
    x_name = "MCF10A_CTCF", y_name = "MCF10CA1_CTCF")

ssvSignalScatterplot(CTCF_in_10a_profiles_gr,
```

```
    x_name = "MCF10A_CTCF", y_name = "MCF10AT1_CTCF",
    value_function = median) + labs(title = "median FE in regions")

ssvSignalScatterplot(CTCF_in_10a_profiles_gr,
    x_name = "MCF10A_CTCF", y_name = "MCF10AT1_CTCF",
    plot_type = "volcano")

ssvSignalScatterplot(CTCF_in_10a_profiles_gr,
    x_name = "MCF10A_CTCF", y_name = "MCF10AT1_CTCF",
    plot_type = "volcano", show_help = TRUE)
```

`test_peaks` *4 random peaks for single-end data and 4 control regions 30kb downstream from each peak.*

Description

matches `system.file("extdata/test_peaks.bam",package = "seqsetvis")`

Format

GRanges length 8

Details

this is included only for testing ssvFetchBam functions.

`viewGRangesWinSample_dt` *get a windowed sampling of score_gr*

Description

This method is appropriate when all GRanges in qgr are identical width and when it is practical to use a window_size smaller than features in genomic signal. For instance, when retrieving signal around peaks or promoters this method maintains a fixed genomic scale across regions. This allows meaingful comparison of peak widths can be made.

Usage

```
viewGRangesWinSample_dt(score_gr, qgr, window_size, attrib_var = "score",
  fill_value = 0, anchor = c("center", "center_unstranded", "left",
  "left_unstranded")[1])
```

Arguments

score_gr	GRanges with a "score" metadata column.
qgr	regions to view by window.
window_size	qgr will be represented by value from score_gr every window_size bp.
attrib_var	character name of attribute to pull data from. Default is "score", compatible with with bigWigs or bam coverage.
fill_value	numeric or character value to use where queried regions are empty. Default is 0 and appropriate for both calculated coverage and bedgraph/bigwig like files. Will automatically switch to "MISSING" if data is guessed to be qualitative.
anchor	character. controls how x value is derived from position for each region in qgr. 0 may be the left side or center. If not unstranded, x coordinates are flipped for (-) strand. One of c("center", "center_unstranded", "left", "left_unstranded"). Default is "center".

Details

Summarizes score_gr by grabbing value of "score" every window_size bp. Columns in output data.table are: standard GRanges columns: seqnames, start, end, width, strand id - matched to names(score_gr). if names(score_gr) is missing, added as 1:length(score_gr) y - value of score from score_gr x - relative bp position

Value

data.table that is GRanges compatible

Examples

```
bam_file = system.file("extdata/test.bam",
    package = "seqsetvis")
qgr = CTCF_in_10a_overlaps_gr[1:5]
qgr = GenomicRanges::resize(qgr, width = 500, fix = "center")
bam_gr = seqsetvis:::fetchBam(bam_file, qgr)
bam_dt = viewGRangesWinSample_dt(bam_gr, qgr, 50)

if(Sys.info()['sysname'] != "Windows"){
    bw_file = system.file("extdata/MCF10A_CTCF_FE_random100.bw",
        package = "seqsetvis")
    bw_gr = rtracklayer::import.bw(bw_file, which = qgr)
    bw_dt = viewGRangesWinSample_dt(bw_gr, qgr, 50)
}
```

---

viewGRangesWinSummary_dt

Summarizes signal in bins. The same number of bins per region in qgr is used and widths can vary in qgr, in contrast to viewGRangesWinSample_dt *where width must be constant across regions.*

Description

This function is most appropriate where features are expected to vary greatly in size and feature boundaries are important, ie. gene bodies, enhancers or TADs.

Usage

```
viewGRangesWinSummary_dt(score_gr, qgr, n_tiles = 100,
  attrib_var = "score", attrib_type = NULL, fill_value = 0,
  anchor = c("center", "center_unstranded", "left",
  "left_unstranded")[1], summary_FUN = stats::weighted.mean)
```

Arguments

score_gr	GRanges with a "score" metadata column.
qgr	regions to view by window.
n_tiles	numeric >= 1, the number of tiles to use for every region in qgr.
attrib_var	character name of attribute to pull data from. Default is "score", compatible with with bigWigs or bam coverage.
attrib_type	one of NULL, qualitative or quantitative. If NULL will attempt to guess by casting attrib_var attribute to character or factor. Default is NULL.
fill_value	numeric or character value to use where queried regions are empty. Default is 0 and appropriate for both calculated coverage and bedgraph/bigwig like files. Will automatically switch to "MISSING" if data is guessed to be qualitative.
anchor	character. controls how x value is derived from position for each region in qgr. 0 may be the left side or center. If not unstranded, x coordinates are flipped for (-) strand. One of c("center", "center_unstranded", "left", "left_unstranded"). Default is "center".
summary_FUN	function. used to aggregate score by tile. must accept x=score and w=width numeric vectors as only arguments. default is weighted.mean. limma::weighted.median is a good alternative.

Details

Columns in output data.table are: standard GRanges columns: seqnames, start, end, width, strand id - matched to names(score_gr). if names(score_gr) is missing, added as 1:length(score_gr) y - value of score from score_gr x - relative bp position

Value

data.table that is GRanges compatible

Examples

```
bam_file = system.file("extdata/test.bam",
    package = "seqsetvis")
qgr = CTCF_in_10a_overlaps_gr[1:5]
# unlike viewGRangesWinSample_dt, width is not fixed
# qgr = GenomicRanges::resize(qgr, width = 500, fix = "center")
bam_gr = seqsetvis:::fetchBam(bam_file, qgr)
bam_dt = viewGRangesWinSummary_dt(bam_gr, qgr, 50)

if(Sys.info()['sysname'] != "Windows"){
```

```
    bw_file = system.file("extdata/MCF10A_CTCF_FE_random100.bw",
        package = "seqsetvis")
    bw_gr = rtracklayer::import.bw(bw_file, which = qgr)
    bw_dt = viewGRangesWinSummary_dt(bw_gr, qgr, 50)
}
```

Index