Package 'spp'

May 30, 2019

Type Package

Title ChIP-Seq Processing Pipeline

Version 1.16.0

Author Peter K

Depends R (>= 3.3.0), Rcpp

Imports Rsamtools, caTools, parallel, graphics, stats

Suggests methods

LinkingTo Rcpp, BH (>= 1.66)

OS_type unix

Maintainer Peter Kharchenko <spppackage.maintenance@gmail.com>

Description Analysis of ChIP-
seq and other functional sequencing data [Kharchenko PV (2008) <DOI:10.1038/nbt.1508>].

License GPL-2

LazyLoad yes

Note revised for compliance with CRAN by K.Pal and C.M.Livi

NeedsCompilation yes

Repository CRAN

Date/Publication 2019-05-30 13:20:04 UTC

R topics documented:

spp-package 2
add.broad.peak.regions 6
densum 7
find.binding.positions 8
get.binding.characteristics 12
get.broad.enrichment.clusters 13
get.conservative.fold.enrichment.profile 14
get.conservative.fold.enrichment.profile2 16
get.mser 18

get.mser.interpolation . . . 19
get.smoothed.enrichment.mle . . . 21
get.smoothed.enrichment.mle2 . . . 22
get.smoothed.tag.density . . . 24
output.binding.results . . . 25
points_within . . . 26
read.arachne.tags . . . 28
read.bam.tags . . . 29
read.bin.maqmap.tags . . . 29
read.bowtie.tags . . . 30
read.eland.tags . . . 30
read.helicos.tags . . . 31
read.maqmap.tags . . . 33
read.meland.tags . . . 33
read.short.arachne.tags . . . 34
read.tagalign.tags . . . 35
remove.local.tag.anomalies . . . 37
select.informative.tags . . . 38
write.broadpeak.info . . . 39
write.narrowpeak.binding . . . 39
writewig . . . 40

Index **41**

---

spp-package *ChIP-seq (Solexa) Processing Pipeline*

Description

A set of routines for reading short sequence alignments, calculating tag density, estimates of statistically significant enrichment/depletion along the chromosome, identifying point binding positions (peaks), and characterizing saturation properties related to sequencing depth.

Details

Package:	spp
Type:	Package
Version:	1.11
Date:	2012-06-20
License:	What license is it under?
LazyLoad:	yes

See example below for typical processing sequence.y

Author(s)

Peter Kharchenko <peter.kharchenko@post.harvard.edu>

References

Kharchenko P., Tolstorukov M., Park P. "Design and analysis of ChIP-seq experiments for DNA-binding proteins." Nature Biotech. doi:10.1038/nbt.1508

Examples

```
## Not run:
  # load the library
  library(spp);

  ## The following section shows how to initialize a cluster of 8 nodes for parallel processing
  ## To enable parallel processing, uncomment the next three lines,
  #and comment out "cluster<-NULL";
  ## see "snow" package manual for details.
  #library(snow)
  #cluster <- makeCluster(2);
  #invisible(clusterCall(cluster,source,"routines.r"));
  cluster <- NULL;



  # read in tag alignments
  chip.data <- read.eland.tags("chip.eland.alignment");
  input.data <- read.eland.tags("input.eland.alignment");

  # get binding info from cross-correlation profile
  # srange gives the possible range for the size of the protected region;
  # srange should be higher than tag length; making the upper
  #boundary too high will increase calculation time
  #
  # bin - bin tags within the specified number of basepairs to speed up calculation;
  # increasing bin size decreases the accuracy of the determined parameters
  binding.characteristics <- get.binding.characteristics(chip.data,
    srange=c(50,500),
    bin=5,
    cluster=cluster);


  # plot cross-correlation profile
  pdf(file="example.crosscorrelation.pdf",width=5,height=5)
  par(mar = c(3.5,3.5,1.0,0.5), mgp = c(2,0.65,0), cex = 0.8);
  plot(binding.characteristics$cross.correlation,
    type='l',
    xlab="strand shift",
    ylab="cross-correlation");
  abline(v=binding.characteristics$peak$x,lty=2,col=2)
  dev.off();
```

```
# select informative tags based on the binding characteristics
chip.data <- select.informative.tags(chip.data,binding.characteristics);
input.data <- select.informative.tags(input.data,binding.characteristics);

# restrict or remove positions with anomalous number of tags relative
# to the local density
chip.data <- remove.local.tag.anomalies(chip.data);
input.data <- remove.local.tag.anomalies(input.data);


# output smoothed tag density (subtracting re-scaled input) into a WIG file
# note that the tags are shifted by half of the peak separation distance
smoothed.density <- get.smoothed.tag.density(chip.data,
  control.tags=input.data,
  bandwidth=200,
  step=100,
  tag.shift=round(binding.characteristics$peak$x/2));

writewig(smoothed.density,
  "example.density.wig","Example smoothed, background-subtracted tag density");

rm(smoothed.density);

# output conservative enrichment estimates
# alpha specifies significance level at which confidence intervals will be estimated
enrichment.estimates <- get.conservative.fold.enrichment.profile(chip.data,
  input.data,
  fws=2*binding.characteristics$whs,
  step=100,
  alpha=0.01);

writewig(enrichment.estimates,
  "example.enrichment.estimates.wig",
  "Example conservative fold-enrichment/depletion estimates shown on log2 scale");
rm(enrichment.estimates);


# binding detection parameters
# desired FDR. Alternatively, an E-value can be supplied to the
#method calls below instead of the fdr parameter
fdr <- 1e-2;
# the binding.characteristics contains the optimized half-size for binding detection window
detection.window.halfsize <- binding.characteristics$whs;

# determine binding positions using wtd method
bp <- find.binding.positions(signal.data=chip.data,
  control.data=input.data,
  fdr=fdr,
  method=tag.wtd,
  whs=detection.window.halfsize,
  cluster=cluster)

# alternatively determined binding positions using lwcc
```

```
# method (note: this takes longer than wtd)
# bp <- find.binding.positions(signal.data=chip.data,control.data=input.data,
#fdr=fdr,method=tag.lwcc,whs=detection.window.halfsize,cluster=cluster)

print(paste("detected",sum(unlist(lapply(bp$npl,function(d) length(d$x)))),"peaks"));

# output detected binding positions
output.binding.results(bp,"example.binding.positions.txt");


# ------------------------------------------------------------------------------------------
# the set of commands in the following section illustrates methods for saturation analysis
# these are separated from the previous section, since they are highly CPU intensive
# ------------------------------------------------------------------------------------------

# determine MSER
# note: this will take approximately 10-15x the amount of time the initial
# binding detection did
# The saturation criteria here is 0.99 consistency in the set of binding
# positions when adding 1e5 tags.
# To ensure convergence the number of subsampled chains (n.chains) should be higher (80)
mser <- get.mser(chip.data,
  input.data,
  step.size=1e5,
  test.agreement=0.99,
  n.chains=8,
  cluster=cluster,
  fdr=fdr,
  method=tag.wtd,
  whs=detection.window.halfsize)

print(paste("MSER at a current depth is",mser));

# note: an MSER value of 1 or very near one implies that the set of
# detected binding positions satisfies saturation criteria without
# additional selection by fold-enrichment ratios. In other words,
# the dataset has reached saturation in a traditional sense (absolute saturation).

# interpolate MSER dependency on tag count
# note: this requires considerably more calculations than the previous
# steps (~ 3x more than the first MSER calculation)
# Here we interpolate MSER dependency to determine a point at which MSER of 2 is reached
# The interpolation will be based on the difference in MSER at the
# current depth, and a depth at 5e5 fewer tags (n.steps=6);
# evaluation of the intermediate points is omitted here to
# speed up the calculation (excluded.steps parameter)
# A total of 7 chains is used here to speed up calculation,
# whereas a higher number of chains (50) would give good convergence
msers <- get.mser.interpolation(chip.data,
  input.data,
  step.size=1e5,
  test.agreement=0.99,
  target.fold.enrichment=2,
```

```
  n.chains=7,
  n.steps=6,
  excluded.steps=c(2:4),
  cluster=cluster,
  fdr=fdr,
  method=tag.wtd,
  whs=detection.window.halfsize)

print(paste("predicted sequencing depth =",
  round(unlist(lapply(msers,function(x) x$prediction))/1e6,5)," million tags"))


# note: the interpolation will return NA prediction if the dataset
# has reached absolute saturation at the current depth.
# note: use return.chains=T to also calculated random chains
# returned under msers$chains field) - these can be passed back as
#       "get.mser.interpolation( ..., chains=msers$chains)" to calculate
#       predictions for another target.fold.enrichment value
#        without having to recalculate the random chain predictions.

## stop cluster if it was initialized
#stopCluster(cluster);



## End(Not run)
```

`add.broad.peak.regions` *Calculate chromosome-wide profiles of smoothed tag density*

Description

Looks for broader regions of enrichment associated with the determined peak positions, adds them to the $npl data as $rs, $re columns.

Usage

```
add.broad.peak.regions(chip.tags,input.tags,bp,window.size=500,z.thr=2)
```

Arguments

`chip.tags`	signal chromosome tag coordinate vectors (e.g. output of `select.informative.tags`
`input.tags`	optionall control (input) tags
`bp`	output of find.binding.positions call
`window.size`	window size to be used in calculating enrichment
`z.thr`	Z-score corresponding to the Poisson ratio threshold used to flag significantly enriched windows

Value

A structure identical to bp (binding.postions) with two additional columns added (rs and re) corresponding to start and end of the associated significantly enriched region. If no region was associated with a particular peak, NAs values are reported.

`densum` *Do Something*

Description

Densum

Usage

```
densum(vin, bw = 5, dw = 3, match.wt.f = NULL, return.x = T,
       from = min(vin), to = max(vin), step = 1, new.code = T)
```

Arguments

`vin`	Parameter
`bw`	Parameter
`dw`	Parameter
`match.wt.f`	Parameter
`return.x`	Parameter
`from`	Parameter
`to`	Parameter
`step`	Parameter
`new.code`	Parameter

Value

Some sum

Examples

```
## Not run:
##---- Should be DIRECTLY executable !! ----
##-- ==>  Define data, use random,
##--or do  help(data=index)  for the standard data sets.

## The function is currently defined as
function (vin, bw = 5, dw = 3, match.wt.f = NULL, return.x = T,
    from = min(vin), to = max(vin), step = 1, new.code = T)
{
    tc <- table(vin[vin >= from & vin <= to])
```

```
    pos <- as.numeric(names(tc))
    storage.mode(pos) <- "double"
    tc <- as.numeric(tc)
    storage.mode(tc) <- "double"
    n <- length(pos)
    if (!is.null(match.wt.f)) {
        tc <- tc * match.wt.f(pos)
    }
    rng <- c(from, to)
    if (rng[1] < 0) {
        stop("range extends into negative values")
    }
    if (range(pos)[1] < 0) {
        stop("position vector contains negative values")
    }
    storage.mode(n) <- storage.mode(rng) <- storage.mode(bw)
          <- storage.mode(dw) <- storage.mode(step) <- "integer"
    spos <- rng[1]
    storage.mode(spos) <- "double"
    dlength <- floor((rng[2] - rng[1])/step) + 1
    if (dlength < 1) {
        stop("zero data range")
    }
    if (new.code) {
        storage.mode(step) <- storage.mode(dlength)
              <- storage.mode(bw) <- storage.mode(dw) <- "integer"
        dout <- .Call("ccdensum", pos, tc, spos, bw, dw, dlength,
            step)
    }
    else {
        dout <- numeric(dlength)
        storage.mode(dout) <- "double"
        storage.mode(dlength) <- "integer"
        .C("cdensum", n, pos, tc, spos, bw, dw, dlength, step,
            dout)
    }
    if (return.x) {
        return(list(x = c(rng[1], rng[1] + step * (dlength -
            1)), y = dout, step = step))
    }
    else {
        return(dout)
    }
  }

## End(Not run)
```

`find.binding.positions` *Determine significant point protein binding positions (peaks)*

Description

Given the signal and optional control (input) data, determine location of the statistically significant point binding positions. If the control data is not provided, the statistical significance can be assessed based on tag randomization. The method also provides options for masking regions exhibiting strong signals within the control data.

Usage

```
find.binding.positions(signal.data,  f=1,e.value = NULL, fdr = NULL, masked.data = NULL,
  control.data = NULL, whs = 200, min.dist = 200, window.size = 4e+07, cluster = NULL,
  debug = T, n.randomizations = 3, shuffle.window = 1, min.thr = 2, topN = NULL,
  tag.count.whs = 100, enrichment.z = 2, method = tag.wtd, tec.filter = T,
  tec.window.size = 10000, tec.z = 5, tec.masking.window.size=tec.window.size,
  tec.poisson.z=5,tec.poisson.ratio=5, tec = NULL, n.control.samples = 1,
  enrichment.scale.down.control =F, enrichment.background.scales = c(1, 5, 10),
  use.randomized.controls = F, background.density.scaling = T,
  mle.filter = F,min.mle.threshold = 1, ...)
```

Arguments

	~~ tag data ~~
	signal tag vector list
control.data	optional control (input) tag vector list
f	Fraction of signal read subsampled. Default=1, i.e. no subsampling ~~ position stringency criteria ~~
e.value	E-value defining the desired statistical significance of binding positions.
fdr	FDR defining statistical significance of binding positions
topN	instead of determining statistical significance thresholds, return the specified number of highest-scoring positions
	~~ other params ~~
whs	window half-sized that should be used for binding detection (e.g. determined from cross-correlation profiles)
masked.data	optional set of coordinates that should be masked (e.g. known non-unique regions)
min.dist	minimal distance that must separate detected binding positions. In case multiple binding positions are detected within such distance, the position with the highest score is returned.
window.size	size of the window used to segment the chromosome during calculations to reduce memory usage.
cluster	optional snow cluster to parallelize the processing on
debug	debug mode, whether to print debug messages
min.thr	minimal score requirement for a peak

`background.density.scaling`
: If TRUE, regions of significant tag enrichment will be masked out when calculating size ratio of the signal to control datasets (to estimate ratio of the background tag density). If FALSE, the dataset ratio will be equal to the ratio of the number of tags in each dataset.

`tec`
: tec

`n.control.samples`
: n.control.samples

`enrichment.scale.down.control`
: enrichment.scale.down.control

`...`
: additional parameters should be the same as those passed to the `lwcc.prediction`

~~ randomized controls ~~

`n.randomizations`
: number of tag randomziations that should be performed (when the control data is not provided)

`use.randomized.controls`
: Use randomized tag control, even if `control.data` is supplied.

`shuffle.window`
: during tag randomizations, tags will be split into groups of `shuffle.window` and will be maintained together throughout the randomization.

~~ fold-enrichment confidence intervals ~~

`tag.count.whs`
: half-size of a window used to assess fold enrichment of a binding position

`enrichment.z`
: Z-score used to define the significance level of the fold-enrichment confidence intervals

`enrichment.background.scales`
: In estimating the peak fold-enrichment confidence intervals, the background tag density is estimated based on windows with half-sizes of `2*tag.count.whs*enrichment.background.s`

`method`
: either `tag.wtd` for WTD method, or `tag.lwcc` for MTC method

`mle.filter`
: If turned on, will exclude predicted positions whose MLE enrichment ratio (for any of the background scales) is below a specified min.mle.threshold

`min.mle.threshold`
: MLE enrichment ratio threshold that each predicted position must exceed if mle.filter is turned on.

~~ masking regions of significant control enrichment ~~

`tec.filter`
: Whether to mask out the regions exhibiting significant enrichment in the control data in doing other calculations. The regions are identified using Poisson statistics within sliding windows, either relative to the scaled signal (tec.z), or relative to randomly-distributed expectation (tec.poisson.z).

`tec.window.size`
: size of the window used to determine significantly enrichent control regions

`tec.masking.window.size`
: size of the window used to mask the area around significantly enrichent control regions

`tec.z`
: Z-score defining statistical stringency by which a given window is determined to be significantly higher in the input than in the signal, and masked if that is the case.

`tec.poisson.z` Z-score defining statistical stringency by which a given window is determined to be significantly higher than the tec.poisson.ratio above the expected uniform input background.

`tec.poisson.ratio` Fold ratio by which input must exceed the level expected from the uniform distribution.

Value

- npl A per-chromosome list containing data frames describing determined binding positions. Column description
  - x: position
  - y: score
  - evalue: E-value
  - fdr: FDR. For peaks higher than the maximum control peak,the highest dataset FDR is reported
  - enr: lower bound of the fold-enrichment ratio confidence interval. This is the estimate determined using scale of 1. Estimates corresponding to higher scales are returned in other enr columns with scale appearing in the name.
  - enr.mle: enrichment ratio maximum likely estimate
- thr: info on the chosen statistical threshold of the peak scores

Examples

```
## Not run:
# find binding positions using WTD method, 200bp half-window size,
# control data, 1
bp <-find.binding.positions(signal.data=chip.data,
    control.data=input.data,
    fdr=0.01,
    method=tag.wtd,
    whs=200);

# find binding positions using MTC method, using 5 tag randomizations,
#  keeping pairs of tag positions together (shuffle.window=2)
bp <- find.binding.positions(signal.data=chip.data,
  control.data=input.data,
  fdr=0.01,method=tag.lwcc,
    whs=200,
    use.randomized.controls=T,
    n.randomizations=5,
    shuffle.window=2)

# print out the number of determined positions
print(paste("detected",sum(unlist(lapply(bp$npl,function(d) length(d$x)))),"peaks"));


## End(Not run)
```

get.binding.characteristics *Calculate characteristics of observed DNA-binding signal from cross-correlation profiles*

Description

The methods calculates strand cross-correlation profile to determine binding peak separation distance and approximate window size that should be used for binding detection. If quality scores were given for the tags, which quality bins improve the cross-correlation pattern.

Usage

```
get.binding.characteristics(data, srange = c(50, 500), bin = 5,
  cluster = NULL, debug = F, min.tag.count = 1000,
  acceptance.z.score = 3, remove.tag.anomalies = T,
  anomalies.z = 5,accept.all.tags=F)
```

Arguments

data	Tag/quality data: output of read.eland.tags or similar function
srange	A range within which the binding peak separation is expected to fall. Should be larger than probe size to avoid artifacts.
bin	Resolution (in basepairs) at which cross-corrrelation should be calculated. bin=1 is ideal, but takes longer to calculate.
cluster	optional snow cluster for parallel processing
debug	whether to print debug messages
min.tag.count	minimal number of tags on the chromosome to be considered in the cross-correlation calculations
acceptance.z.score	A Z-score used to determine if a given tag quality bin provides significant improvement to the strand cross-correlation
remove.tag.anomalies	Whether to remove singular tag count peaks prior to calculation. This is recommended, since such positions may distort the cross-correlation profile and increase the necessary computational time.
anomalies.z	Z-score for determining if the number of tags at a given position is significantly higher about background, and should be considered an anomaly.
accept.all.tags	Whether tag alignment quality calculations should be skipped and all available tags should be accepted in the downstream analysis.

Value

`cross.correlation`
: Cross-correlation profile as an $x/$y data.frame

`peak`
: Position ($x) and height ($y) of automatically detected cross-correlation peak.

`whs`
: Optimized window half-size for binding detection (based on the width of the cross-correlation peak)

`quality.bin.acceptance`
: A list structure, describing the effect of inclusion of different tag quality bins on cross-correlation, and a resolution on which bins should be considered.

- informative.bins: A boolean vector indicating whether the inclusion of tags from the tag quality bin specified in the name attribute signiificantly increases cross-correlation profile near the peak.
- quality.cc: A list giving the cross-correlation profile after the inclusion of the tags from different quality bins

---

`get.broad.enrichment.clusters` *Determine broad clusters of enrichment*

Description

Scan chromosomes with a pre-defined window size, comparing scaled ChIP and input tag coutns to see if their ratio exceeds that expected from a Poisson process (normalized for dataset size).

Usage

```
get.broad.enrichment.clusters(signal.data,
  control.data,
  window.size=1e3,
  z.thr=3,
  tag.shift=146/2,
  background.density.scaling = F,
  ...)
```

Arguments

`signal.data`
: chip.tags, foreground tag vector list

`control.data`
: input.tags, background tag vector list

`window.size`
: window size to be used for tag counting

`z.thr`
: Z-score to be used as a significance threshold

`tag.shift`
: number of base pairs by which positive and negative tag coordinates should be shifted towards eachother (half of binding peak separation distance)

background.density.scaling
: If TRUE, regions of significant tag enrichment will be masked out when calculating size ratio of the signal to control datasets (to estimate ratio of the background tag density). If FALSE, the dataset ratio will be equal to the ratio of the number of tags in each dataset.

...
: additional parameters should be the same as those passed to the find.significantly.enriched.regior

Value

A list of elements corresponding to chromosomes, with each element being an $s/$e/$rv data.frame giving the starting, ending positions and the log2 enrichment estimate for that region.

---

get.conservative.fold.enrichment.profile

Estimate minimal fold enrichment/depletion along the chromosomes

---

Description

The method provides a statistical assessment of enrichment/depletion along the chromosomes. To assess tag density enrichment/depletion, a sliding window of a specified size (fws) is used to calculate the density of the foreground tags (ftl). Multiple, typically larger windows are used to estimate background tag (btl) density around the same location. The densities are compared as ratios of two Poisson processes to estimate lower bound of foreground enrichment, or upper bound of foreground depletion. If multiple window sizes were used to estimate the background tag density, the most conservative one is chosen for each point.

Usage

```
get.conservative.fold.enrichment.profile(ftl,
  btl,
  fws,
  bwsl = c(1, 5, 25, 50) * fws,
  step = 50,
  tag.shift = 146/2,
  alpha = 0.05,
  use.most.informative.scale = F,
  quick.calculation = T,
  background.density.scaling = T,
  bg.weight = NULL,
  posl= NULL,
  return.mle = F)
```

Arguments

ftl	foreground tag vector list
btl	background tag vector list
fws	foreground window size

bwsl	background window scales. The size(s) of background windows will be fws*bwsl.
step	spacing between positions at which the enrichment/depletion is evaluated
tag.shift	number of basepairs by which positive and negative tag coordinates should be shifted towards eachother (half of binding peak separation distance)
alpha	desired level of statistical significance
use.most.informative.scale	for each position, instead of evaluating enrichment ratio bounds for all background window scales, choose the one with the highest observed density to speed up the calculations
quick.calculation	Use square root transformation method instead of a Bayesian method. This speeds up the caclulation considerably and is turned on by default.
background.density.scaling	If TRUE, regions of significant tag enrichment will be masked out when calculating size ratio of the signal to control datasets (to estimate ratio of the background tag density). If FALSE, the dataset ratio will be equal to the ratio of the number of tags in each dataset.
bg.weight	optional weight by which the background density should be multipled for scaling. If not supplied, the weight is calculated based on the ratio of the reduced ChIP to input dataset sizes.
posl	posl
return.mle	return.mle

Value

A list of elements corresponding to chromosomes, with each element being an $x/$y data.frame giving the position and the log2 conservative estimate of enrichment/depletion fold ratios around that position. Use writewig to output the structure to a WIG file.

References

R.M.Price, D.G. Bonett "Estimating the ratio fo two Poisson rates", Comp. Stat & Data Anal. 32(2000) 345

See Also

get.smoothed.tag.density

Examples

```
## Not run:
    enrichment.estimates <- get.conservative.fold.enrichment.profile(chip.data,
      input.data,
      fws=2*binding.characteristics$whs,
      step=100,
      alpha=0.01);
    writewig(enrichment.estimates,
```

```
        "example.enrichment.estimates.wig",
        "Example conservative fold-enrichment/depletion estimates shown on log2 scale");

## End(Not run)
```

`get.conservative.fold.enrichment.profile2` *Return Conservative fold enrichment profile controlling for input and a single background scale*

Description

Returns a conservative upper/lower bound profile (log2) given signal tag list, background tag list and window scales controlling for input, and supporting only a single background scale.

Novel version of get.conservative.fold.enrichment.profile() supporting a single background scale.

Usage

```
get.conservative.fold.enrichment.profile2(ftl1, ftl2, btl1, btl2, fws,
            bws = 1 * fws, step = 50, tag.shift = 146/2,
            alpha = 0.05, background.density.scaling = T,
            bg.weight1 = NULL, bg.weight2 = NULL,
            posl = NULL, return.mle = F)
```

Arguments

`ftl1`	Parameter
`ftl2`	Parameter
`btl1`	Parameter
`btl2`	Parameter
`fws`	Parameter
`bws`	Parameter
`step`	Parameter
`tag.shift`	Parameter
`alpha`	Parameter
`background.density.scaling`	Parameter
`bg.weight1`	Parameter
`bg.weight2`	Parameter
`posl`	Parameter
`return.mle`	Parameter

Value

A list of elements corresponding to chromosomes, with each element being an $x/$y data.frame giving the position and associated log2 signal/control enrichment estimate.

See Also

get.smoothed.enrichment.mle

Examples

```
## Not run:
##---- Should be DIRECTLY executable !! ----
##-- ==>  Define data, use random,
##--or do  help(data=index)  for the standard data sets.

## The function is currently defined as
function (ftl1, ftl2, btl1, btl2, fws, bws = 1 * fws, step = 50,
    tag.shift = 146/2, alpha = 0.05, background.density.scaling = T,
    bg.weight1 = NULL, bg.weight2 = NULL, posl = NULL, return.mle = F)
{
    ftl1 <- ftl1[unlist(lapply(ftl1, length)) > 2]
    chrl <- names(ftl1)
    names(chrl) <- chrl
    if (!is.null(posl)) {
        chrl <- chrl[chrl %in% names(posl)]
    }
    if (is.null(bg.weight1)) {
        bg.weight1 <- dataset.density.ratio(ftl1, btl1,
                      background.density.scaling = background.density.scaling)
    }
    if (is.null(bg.weight2)) {
        bg.weight2 <- dataset.density.ratio(ftl2, btl2,
                      background.density.scaling = background.density.scaling)
    }
    lapply(chrl, function(chr) {
        x <- binomial.proportion.ratio.bounds(abs(ftl1[[chr]] +
            tag.shift), abs(btl1[[chr]] + tag.shift), abs(ftl2[[chr]] +
            tag.shift), abs(btl2[[chr]] + tag.shift), fws = fws,
            bws = bws, step = step, bg.weight1 = bg.weight1,
            bg.weight2 = bg.weight2, alpha = alpha, pos = if (is.null(posl)) {
                NULL
            }
            else {
                posl[[chr]]
            })
        ps <- rep(0, length(x$mle))
        vi <- which(!is.na(x$lb) & x$lb > 0)
        ps[vi] <- x$lb[vi]
        vi <- which(!is.na(x$ub) & x$ub < 0)
        ps[vi] <- x$ub[vi]
        if (is.null(posl)) {
            if (return.mle) {
```

```
                return(data.frame(x = x$x, y = ps, mle = x$mle,
                  lb = x$lb, ub = x$ub))
            }
            else {
                return(data.frame(x = x$x, y = ps))
            }
        }
        else {
            if (return.mle) {
                return(data.frame(x = posl[[chr]], y = ps, mle = x$mle,
                  lb = x$lb, ub = x$ub))
            }
            else {
                return(data.frame(x = posl[[chr]], y = ps))
            }
        }
    })
  }

## End(Not run)
```

`get.mser` *Calculate minimal saturated enrichment fold ratio*

Description

Determine if the dataset has reached absolute saturation, or otherwise find minimal fold enrichment ratio above which the detection of peaks has stabilized enough to meet the saturation criteria.

Usage

```
get.mser(signal.data,
  control.data,
  n.chains = 5,
  step.size = 1e+05,
  chains = NULL,
  cluster = NULL,
  test.agreement = 0.99,
  return.chains = F,
  enrichment.background.scales = c(1),
  n.steps = 1, ...)
```

Arguments

`signal.data`	signal tag vector list
`control.data`	control tag vector list
`n.chains`	number of dataset subsamples to use

step.size	subsampling step describing the saturation criteria. The criteria requires the set of detected binding sites to be stable (as described by the test.agreement param) when the number of tags in the dataset is reduced by step.size. The value can either be an integer above one, in which case it specifies a fixed number of tags, or a real value below one, in which case it specifies the fraction of tags that should be removed (e.g. 0.1 will remove 10
test.agreement	Fraction of the detected peaks that should agree between the full and subsampled datasets.
chains	optional parameter, giving pre-calculated chains
cluster	optional snow cluster to parallelize processing
return.chains	whether subsampled dataset results should be returned as well
enrichment.background.scales	one or multiple window scales at which the background tag density should be assessed. See enrichment.background.scales in find.binding.positions. If multiple scales are provided, multiple MSER estimates will be returned.
...	additional parameters should be the same as those passed to the find.binding.positions
n.steps	n.steps

Value

A single, or multple (if multiple enrichment.background.scales were provided) MSER value. A value of 1 or very close to it implies that the dataset has reached absolute saturation based on the given criteria.

---

get.mser.interpolation

Interpolate MSER dependency on the tag count

Description

MSER generally decreases with increasing sequencing depth. This function interpolates the dependency of MSER on tag counts as a log-log linear function. The log-log fit is used to estimate the depth of sequencing required to reach desired target.fold.enrichment.

Usage

```
get.mser.interpolation(signal.data,
  control.data,
  target.fold.enrichment = 5,
  n.chains = 10,
  n.steps = 6,
  step.size = 1e+05,
  chains = NULL,
  test.agreement = 0.99,
  return.chains = F,
  enrichment.background.scales = c(1),
  excluded.steps = c(seq(2, n.steps - 2)), ...)
```

Arguments

signal.data	signal chromosome tag vector list
control.data	control chromosome tag vector list
target.fold.enrichment	target MSER for which the depth should be estimated
n.steps	number of steps in each subset chain.
step.size	Either number of tags or fraction of the dataset size, see step.size parameter for get.mser.
test.agreement	Fraction of the detected peaks that should agree between the full and subsampled datasets. See test.agreement parameter for get.mser
n.chains	number of random subset chains
chains	optional structure of pre-calculated chains (e.g. generated by an earlier call with return.chains=T.
return.chains	whether to return peak predictions calculated on random chains. These can be passed back using chains argument to skip subsampling/prediction steps, and just recalculate the depth estimate for a different MSER.
enrichment.background.scales	see enrichment.background.scales parameter for get.mser
excluded.steps	Intermediate subsampling steps that should be excluded from the chains to speed up the calculation. By default, all intermediate steps except for first two and last two are skipped. Adding intermediate steps improves interpolation at the expense of computational time.
...	additional parameters are passed to get.mser

Details

To simulate sequencing growth, the method calculates peak predictions on random chains. Each chain is produced by sequential random subsampling of the original data. The number of steps in the chain indicates how many times the random subsampling will be performed.

Value

Normally reurns a list, specifying for each backgroundscale:

prediction	estimated sequencing depth required to reach specified target MSER
log10.fit	linear fit model, a result of lm() call

If return.chains=T, the above structure is returned under interpolation field, along with chains field containing results of find.binding.positions calls on subsampled chains.

`get.smoothed.enrichment.mle` *Calculate chromosome-wide profiles of smoothed enrichment estimate*

Description

Given signal and control tag positions, the method calculates log2 signal to control enrichment esimates (maximum likelihood) for each chromosome, based on the smoothed tag density profile (see get.smoothed.tag.density).

Usage

```
get.smoothed.enrichment.mle(signal.tags,
  control.tags,
  tag.shift = 146/2,
  background.density.scaling = F,
  pseudocount = 1,
  bg.weight = NULL,
  rngl = NULL,
  chrl = NULL,
  ...)
```

Arguments

`signal.tags`	signal chromosome tag coordinate vectors (e.g. output of `select.informative.tags`
`control.tags`	control (input) tags
`tag.shift`	number of base pairs by which positive and negative tag coordinates should be shifted towards eachother (half of binding peak separation distance)
`background.density.scaling`	background.density.scaling
`pseudocount`	pseudocount value to be added to tag density - defaults to 1
`bg.weight`	optional weight by which the background density should be multipled for scaling. If not supplied, the weight is calculated based on the ratio of the reduced ChIP to input dataset sizes.
`rngl`	rngl
`chrl`	chrl
`...`	additional parameters should be the same as those passed to the `get.smoothed.tag.density`, such as for example bandwidth (default value 150) and step (default value 50). see appropriate reference for `get.smoothed.tag.density` for details.

Value

A list of elements corresponding to chromosomes, with each element being an $x/$y data.frame giving the position and associated log2 signal/control enrichment estimate.

See Also

writewig

Examples

```
## Not run:
  # get smoothed enrichment estimate profile using 500bp bandwidth at
  # 50bp steps
  smoothed.M <- get.smoothed.enrichment.mle(chip.data,bandwidth=500,step=50);
  writewig(smoothed.M,"example.smoothedM.wig","Example smoothed log2 intensity ratio estimate");

## End(Not run)
```

---

get.smoothed.enrichment.mle2

Calculate background input controlled chromosome-wide profiles of smoothed enrichment estimate

---

Description

Given signal and control tag positions, the method calculates log2 signal to control enrichment esimates (maximum likelihood) for each chromosome, based on the smoothed tag density profile (see get.smoothed.tag.density).

Usage

```
get.smoothed.enrichment.mle2(signal.tags1, control.tags1,
                             signal.tags2, control.tags2,
                             tag.shift = 146/2, background.density.scaling = F,
                             pseudocount = 1, bg.weight1 = NULL,
                             bg.weight2 = NULL,
                             rngl = NULL, chrl = NULL, ...)
```

Arguments

signal.tags1	Parameter
control.tags1	Parameter
signal.tags2	Parameter
control.tags2	Parameter
tag.shift	Parameter
background.density.scaling	Parameter
pseudocount	Parameter
bg.weight1	Parameter
bg.weight2	Parameter

rngl	Parameter
chrl	Parameter
...	Parameter

Value

A list of elements corresponding to chromosomes, with each element being an $x/$y data.frame giving the position and associated log2 signal/control enrichment estimate.

See Also

get.smoothed.enrichment.mle

Examples

```
## Not run:
##---- Should be DIRECTLY executable !! ----
##-- ==>  Define data, use random,
##--or do  help(data=index)  for the standard data sets.

## The function is currently defined as
function (signal.tags1, control.tags1, signal.tags2, control.tags2,
    tag.shift = 146/2, background.density.scaling = F, pseudocount = 1,
    bg.weight1 = NULL, bg.weight2 = NULL, rngl = NULL, chrl = NULL,
    ...)
{
    if (is.null(chrl)) {
        chrl <- intersect(names(signal.tags1), names(signal.tags2))
        names(chrl) <- chrl
    }
    if (is.null(rngl)) {
        rngl <- lapply(chrl, function(chr) range(c(range(abs(signal.tags1[[chr]] +
            tag.shift)), range(abs(signal.tags2[[chr]] + tag.shift)))))
    }
    else {
        chrl <- names(rngl)
        names(chrl) <- chrl
    }
    ssd1 <- get.smoothed.tag.density(signal.tags1, rngl = rngl,
        ..., scale.by.dataset.size = F)
    ssd2 <- get.smoothed.tag.density(signal.tags2, rngl = rngl,
        ..., scale.by.dataset.size = F)
    csd1 <- get.smoothed.tag.density(control.tags1, rngl = rngl,
        ..., scale.by.dataset.size = F)
    csd2 <- get.smoothed.tag.density(control.tags2, rngl = rngl,
        ..., scale.by.dataset.size = F)
    if (is.null(bg.weight1)) {
        bg.weight1 <- dataset.density.ratio(signal.tags1, control.tags1,
            background.density.scaling = background.density.scaling)
    }
    if (is.null(bg.weight2)) {
        bg.weight2 <- dataset.density.ratio(signal.tags2, control.tags2,
```

```
            background.density.scaling = background.density.scaling)
    }
    cmle <- lapply(chrl, function(chr) {
        d <- ssd1[[chr]]
        d$y <- log2(ssd1[[chr]]$y + pseudocount * bg.weight1) -
            log2(csd1[[chr]]$y + pseudocount) - log2(bg.weight1) -
            log2(ssd2[[chr]]$y + pseudocount * bg.weight2) +
            log2(csd2[[chr]]$y + pseudocount) + log2(bg.weight2)
        return(d)
    })
  }

## End(Not run)
```

`get.smoothed.tag.density` *Calculate chromosome-wide profiles of smoothed tag density*

Description

Given tag positions, the method calculates for each chromosome a tag density profile, smoothed by the Gaussian kernel. If the optional control tags are provided, the difference between ChIP and control tag density is returned.

Usage

```
get.smoothed.tag.density(signal.tags,
  control.tags = NULL,
  bandwidth = 150,
  bg.weight = NULL,
  tag.shift = 146/2,
  step = round(bandwidth/3),
  background.density.scaling = T,
  rngl = NULL,
  scale.by.dataset.size = F)
```

Arguments

`signal.tags`	signal chromosome tag coordinate vectors (e.g. output of `select.informative.tags`
`control.tags`	optional control (input) tags
`bandwidth`	standard deviation of the Gaussian kernel
`bg.weight`	optional weight by which the background density should be multipled for scaling. If not supplied, the weight is calculated based on the ratio of the reduced ChIP to input dataset sizes.
`tag.shift`	Distance by which the positive and negative strand tags should be shifted towards eachother. This normally corresponds to the half of the cross-correlation peak position (e.g. `get.binding.characteristics()`$peak$x/2)

step — The distance between the regularly spaced points for which the values should be calculated.

background.density.scaling — If TRUE, regions of significant tag enrichment will be masked out when calculating size ratio of the signal to control datasets (to estimate ratio of the background tag density). If FALSE, the dataset ratio will be equal to the ratio of the number of tags in each dataset.

rngl — rngl

scale.by.dataset.size — scale.by.dataset.size

Value

A list of elements corresponding to chromosomes, with each element being an $x/$y data.frame giving the position and associated tag density. Use writewig to output the structure to a WIG file.

See Also

writewig

Examples

```
## Not run:
  smoothed.density <- get.smoothed.tag.density(chip.data,
    control.tags=input.data,
    bandwidth=200,
    step=100,
    tag.shift=round(binding.characteristics$peak$x/2));

  writewig(smoothed.density,
    "example.density.wig",
    "Example smoothed, background-subtracted tag density");

## End(Not run)
```

---

output.binding.results

Write out determined binding peaks into a text file table

Description

Writes out determined binding positions into a text file. The file will contain a table with each row corresponding to a detected position, with the following columns

- chr: chromosome or target sequence
- pos: position of detected binding site on the chromosome/sequence
- score: a score reflecting magnitude of the binding

- Evalue: E-value corresponding to the peak magnitude
- FDR: FDR corresponding to the peak magnitude
- enrichment.lb: lower bound of the fold-enrichment ratio
- enrichment.mle: maximum likelihood estimate of the fold-enrichment ratio

Usage

```
output.binding.results(results, filename)
```

Arguments

results	output of the find.binding.positions
filename	file name

`points_within` *Find points within*

Description

points_within substitutes the deprecated function "points.within"

Usage

```
points_within(x, fs, fe, return.list = F, return.unique = F,
              sorted = F, return.point.counts = F, ...)
```

Arguments

x	Parameter
fs	Parameter
fe	Parameter
return.list	Parameter
return.unique	Parameter
sorted	Parameter
return.point.counts	Parameter
...	Parameter

Value

Parameter

Examples

```
## Not run:
##---- Should be DIRECTLY executable !! ----
##-- ==>  Define data, use random,
##--or do  help(data=index)  for the standard data sets.

## The function is currently defined as
function (x, fs, fe, return.list = F, return.unique = F, sorted = F,
    return.point.counts = F, ...)
{
    if (is.null(x) | length(x) < 1) {
        return(c())
    }
    if (!sorted) {
        ox <- rank(x, ties.method = "first")
        x <- sort(x)
    }
    se <- c(fs, fe)
    fi <- seq(1:length(fs))
    fi <- c(fi, -1 * fi)
    fi <- fi[order(se)]
    se <- sort(se)
    storage.mode(x) <- storage.mode(fi) <- storage.mode(se) <- "integer"
    if (return.unique) {
        iu <- 1
    }
    else {
        iu <- 0
    }
    if (return.list) {
        il <- 1
    }
    else {
        il <- 0
    }
    if (return.point.counts) {
        rpc <- 1
    }
    else {
        rpc <- 0
    }
    storage.mode(iu) <- storage.mode(il) <- storage.mode(rpc) <- "integer"
    result <- .Call("points_withinC", x, se, fi, il, iu, rpc)
    if (!sorted & !return.point.counts) {
        result <- result[ox]
    }
    return(result)
  }

## End(Not run)
```

`read.arachne.tags` *Read in Arachne tags*

Description

Read in ARACHNE Tag file

Usage

```
read.arachne.tags(filename, fix.chromosome.names = F)
```

Arguments

`filename`	filename
`fix.chromosome.names`	do we fix chromosome names

Value

A list like element

See Also

Fill in later

Examples

```
## Not run:
##---- Should be DIRECTLY executable !! ----
##-- ==>  Define data, use random,
##--or do  help(data=index)  for the standard data sets.

## The function is currently defined as
function (filename, fix.chromosome.names = F)
{
    tl <- lapply(.Call("read_arachne_long", path.expand(filename)),
        function(d) {
            xo <- order(abs(d$t))
            d$t <- d$t[xo]
            d$n <- d$n[xo]
            d$l <- d$l[xo]
            return(d)
        })
    if (fix.chromosome.names) {
        names(tl) <- gsub("\.fa", "", names(tl))
    }
    return(list(tags = lapply(tl, function(d) d$t), quality = lapply(tl,
        function(d) d$n), length = lapply(tl, function(d) d$l)))
  }
```

```
## End(Not run)
```

`read.bam.tags` *Read BAM alignment file*

Description

Reads in aligned reads from BAM file. Note: no split (non-unique) alignemnts should be reported in the BAM file.

Usage

```
read.bam.tags(filename, read.tag.names = F, fix.chromosome.names = F)
```

Arguments

`filename`	BAM file
`read.tag.names`	Whether the tag names should be read in
`fix.chromosome.names`	Whether to remove ".fa" from the end of the sequence names

Value

`tags`	A vector of 5' tag coordinates, with negative values corresponding to tags mapped to the negative strand.
`quality`	Number of mismatches
`names`	Tag names, if `read.tag.names` was set

`read.bin.maqmap.tags` *Read MAQ binary alignment map file*

Description

Reads in MAQ binary map alignment result file

Usage

```
read.bin.maqmap.tags(filename, read.tag.names = F, fix.chromosome.names = T)
```

Arguments

`filename`	MAQ map output file (binary)
`read.tag.names`	Whether the tag names should be read in
`fix.chromosome.names`	Whether to remove ".fa" from the end of the sequence names

Value

tags	A vector of 5' tag coordinates, with negative values corresponding to tags mapped to the negative strand.
quality	Number of mismatches
names	Tag names, if read.tag.names was set

---

read.bowtie.tags *Read bowtie text alignment output file*

Description

Reads in bowtie alignment results in text format

Usage

```
read.bowtie.tags(filename, read.tag.names = F, fix.chromosome.names = F)
```

Arguments

filename	bowtie text output file
read.tag.names	Whether the tag names should be read in
fix.chromosome.names	Whether to remove ".fa" from the end of the sequence names

Value

tags	A vector of 5' tag coordinates, with negative values corresponding to tags mapped to the negative strand.
quality	Number of mismatches
names	Tag names, if read.tag.names was set

---

read.eland.tags *Read eland output file*

Description

Reads in ELAND output file, returning 5'-end tag coordinates and number of mismatches associated with each mapped tag.

Usage

```
read.eland.tags(filename,
  read.tag.names = F,
  fix.chromosome.names = T,
  max.eland.tag.length = -1,
  extended=F,
  multi = F)
```

Arguments

filename	ELAND output file
read.tag.names	Whether the tag names should be read in
fix.chromosome.names	Whether to remove ".fa" from the end of the sequence names
max.eland.tag.length	Specifies max length of the tag sequence considered by ELAND. This needs to be specified if the tags are longer than the sequences considred by ELAND during alignment.
extended	Whether the file is written out in "extended" format provided in GA pipeline 1.0.
multi	Whether the file is written in "multi" format, showing multiple alignments of the reads

Value

tags	A vector of 5' tag coordinates, with negative values corresponding to tags mapped to the negative strand.
quality	Number of mismatches
names	Tag names, if read.tag.names was set

---

read.helicos.tags *Read in helicos tags*

Description

Read in Helicos tags

Usage

```
read.helicos.tags(filename, read.tag.names = F,
                  fix.chromosome.names = F, include.length.info = T)
```

Arguments

filename	filename
read.tag.names	Read in tag names
fix.chromosome.names	Do we fix chromosome names
include.length.info	include length information

Value

A list like structure

See Also

Fill in later

Examples

```
## Not run:
##---- Should be DIRECTLY executable !! ----
##-- ==>  Define data, use random,
##--or do  help(data=index)  for the standard data sets.

## The function is currently defined as
function (filename, read.tag.names = F, fix.chromosome.names = F,
    include.length.info = T)
{
    if (read.tag.names) {
        rtn <- as.integer(1)
    }
    else {
        rtn <- as.integer(0)
    }
    tl <- lapply(.Call("read_helicostabf", path.expand(filename),
        rtn), function(d) {
        xo <- order(abs(d$t))
        d$t <- d$t[xo]
        d$n <- d$n[xo]
        d$l <- d$l[xo]
        if (read.tag.names) {
            d$s <- d$s[xo]
        }
        return(d)
    })
    if (fix.chromosome.names) {
        names(tl) <- gsub("\.fa", "", names(tl))
    }
    if (read.tag.names) {
        return(list(tags = lapply(tl, function(d) d$t), quality = lapply(tl,
            function(d) d$n), length = lapply(tl, function(d) d$l),
            names = lapply(tl, function(d) d$s)))
```

```
    }
    else {
        return(list(tags = lapply(tl, function(d) d$t), quality = lapply(tl,
            function(d) d$n), length = lapply(tl, function(d) d$l)))
    }
  }

## End(Not run)
```

`read.maqmap.tags` *Read MAQ text alignment output file*

Description

Reads in MAQ alignment results in text format (that results from "maq mapview" command.)

Usage

```
read.maqmap.tags(filename, read.tag.names = F, fix.chromosome.names = T)
```

Arguments

`filename`	MAQ text output file
`read.tag.names`	Whether the tag names should be read in
`fix.chromosome.names`	Whether to remove ".fa" from the end of the sequence names

Value

`tags`	A vector of 5' tag coordinates, with negative values corresponding to tags mapped to the negative strand.
`quality`	Number of mismatches
`names`	Tag names, if `read.tag.names` was set

`read.meland.tags` *Read modified BED tag alignment file that contains variable match length information*

Description

Reads in an extended BED tag alignment file. An example line given below: `49 . U1 . 1 . . 23 chr2 -234567` The line above specifies a 23-bp portion of the tag tag with id 49 was aligned with 1 mismatch to the negative strand of chr2 at position 234567.

Usage

```
read.meland.tags(filename, read.tag.names = F, fix.chromosome.names = T)
```

Arguments

filename	name of the extended BED file
read.tag.names	whether to read in tag names
fix.chromosome.names	whether to remove ".fa" from the sequence name ends.

Value

tags	A vector of 5' tag coordinates, with negative values corresponding to tags mapped to the negative strand.
quality	Quality expressed as a float x.y, where x is tag.length - aligned.tag.portion.length, and y is the number of mismatches (must be less than 10).
names	Tag names, if read.tag.names was set

---

read.short.arachne.tags

Read in ARACHNE short tags

Description

Read in short arachne reads

Usage

```
read.short.arachne.tags(filename, fix.chromosome.names = F)
```

Arguments

filename	filename
fix.chromosome.names	Fix chromosome names

Details

Not necessary

Value

A list like structure

Note

No Notes

Author(s)

Peter Kharchenko

References

spp by Peter Kharchenko

See Also

Nothing to see here

Examples

```
## Not run:
##---- Should be DIRECTLY executable !! ----
##-- ==>  Define data, use random,
##--or do  help(data=index)  for the standard data sets.

## The function is currently defined as
function (filename, fix.chromosome.names = F)
{
    tl <- lapply(.Call("read_arachne", path.expand(filename)),
        function(d) {
            xo <- order(abs(d$t))
            d$t <- d$t[xo]
            d$n <- d$n[xo]
            return(d)
        })
    if (fix.chromosome.names) {
        names(tl) <- gsub("\.fa", "", names(tl))
    }
    return(list(tags = lapply(tl, function(d) d$t), quality = lapply(tl,
        function(d) d$n)))
  }

## End(Not run)
```

---

read.tagalign.tags *Read in tagalign tags*

Description

Fill in later

Usage

```
read.tagalign.tags(filename, fix.chromosome.names = T, fix.quality = T)
```

Arguments

`filename`	Filename of tag file
`fix.chromosome.names`	chromosome names
`fix.quality`	fix quality

Details

...

Value

a list like structure

Note

Needs further editing

Author(s)

Peter K.

References

spp by kharchenko

See Also

Buh!

Examples

```
## Not run:
##---- Should be DIRECTLY executable !! ----
##-- ==>  Define data, use random,
##--or do  help(data=index)  for the standard data sets.

## The function is currently defined as
function (filename, fix.chromosome.names = T, fix.quality = T)
{
    tl <- lapply(.Call("read_tagalign", path.expand(filename)),
        function(d) {
            xo <- order(abs(d$t))
            d$t <- d$t[xo]
            d$n <- d$n[xo]
            if (fix.quality) {
                if (min(d$n) < 0.5) {
                  d$n = ceiling(1000/4^d$n)
                }
                break.vals <- unique(sort(c(0, unique(d$n))))
```

```
                d$n <- length(break.vals) - 1 - cut(d$n, breaks = break.vals,
                  labels = F)
            }
            return(d)
        })
    if (fix.chromosome.names) {
        names(tl) <- gsub("\.fa", "", names(tl))
    }
    return(list(tags = lapply(tl, function(d) d$t), quality = lapply(tl,
        function(d) d$n)))
  }

## End(Not run)
```

`remove.local.tag.anomalies` *Restrict or remove positions with too many tags relative to local background.*

Description

In Solexa ChIP-seq experiments some anomalous positions contain extremely high number of tags at the exact coordinates. The function scans the chromosomes, determining local tag density based on a provided `window.size`, doing two types of corrections: 1. removing all tags from positions that exceed local density by `eliminate.fold`; 2. reducing the tag count at positions exceeding `cap.fold` to the maximal allowed count. The statistical significance of counts exceeding either of these two threshold densities is calculated based on Poisson model, with confidence interval determined by the `z.threshold` Z-score parameter.

Usage

```
remove.local.tag.anomalies(tags,
  window.size = 200,
  eliminate.fold = 10,
  cap.fold = 4,
  z.threshold = 3)
```

Arguments

`tags`	Chromosome-list of tag vectors
`window.size`	Size of the window used to assess local density. Increasing the window size considerably beyond the size of the binding features will result in flattened profiles, with bound positions exhibiting a difference of just 1 tag beyond the background.
`eliminate.fold`	Threshold definining fold-over background density above which the position is considered anomalous and removed completely.
`cap.fold`	Threshold fold-over background density above which the position is capped to the maximum statistically likely given local tag density

`z.threshold` Z-score used to assess significance of a given position exceeding either of the two density thresholds.

Value

A modified chromosome-wise tag vector list.

Note

~~further notes~~ Increasing window.size to very large values will result in flat profiles similar to those described by Zhang et al. "Model-based Analysis of ChIP-Seq (MACS)." Genome Biol. 2008 Sep 17;9(9):R137.

References

~put references to the literature/web site here ~

`select.informative.tags` *Choose informative tags*

Description

For datasets with tag alignment quality information (e.g. number of mismatches for Eland alignments), `get.binding.characteristics` determines whether inclusion of tags from each specific quality bin improves the cross-correlation profile. The present function is then used to actually select these informative tags, discarding all other information, including quality scores that are not used in further processing.

Usage

```
select.informative.tags(data, binding.characteristics)
```

Arguments

`data` Full alignment data (a list with $tags and $quality elements)

`binding.characteristics`
result of a `get.binding.characteristics` call. If NULL value is supplied,all tags will be accepted.

Value

A chromosome-wise tag list. Each element of the list corresponds to a chromosome and is a numeric vector of 5' tag coordinates, with sign designating DNA strand. This form of tag data is used for most of the other processing.

write.broadpeak.info *Write out determined broad enrichment regions using broadPeak format*

Description

Writes out broad regions of enrichment determined by the get.broad.enrichment.clusters method in a broadPeak format.

Usage

```
write.broadpeak.info(bp, fname)
```

Arguments

bp	broadpeak.results, output of the get.broad.enrichment.clusters
fname	file name

write.narrowpeak.binding *Write out determined binding peaks using narrowPeak format*

Description

Writes out determined binding positions into a narrowPeak file. The region will correspond to associated broad enrichment region, if such were added using add.broad.peak.regions method. Otherwise the region size will be determined using margin (which defaults to the window half size that was used to determine binding positions) Note: since v1.13, FDR is written out in -log10() scale.

Usage

```
write.narrowpeak.binding(bd, fname, margin=bd$whs, npeaks)
```

Arguments

bd	output of the find.binding.positions
fname	file name
margin	explicit value of the margin to be used if the broad region information is absent (defaults to peak detection window half-size
npeaks	optionally, limit the output to the specified number of top peaks

writewig *A function to save a list of chromosome-wise x/y data frames into a WIG file format.*

Description

Takes a list that contains an $x and $y data.frame for a number of chromosomes and writes it out to a WIG BED style format.

Usage

```
writewig(dat, fname, feature, threshold = 5, zip = F)
```

Arguments

dat	Chromosome coordinate-value data. dat is a list, each member of a list is a data frame with $x and $y columns containing chromosome positions and associated values. The names of the list elements correspond to the chromosomes.
fname	Filename to which the output should be written
feature	Data description to be incorporated into the WIG header
threshold	Optional threshold to be saved in the WIG file
zip	Wheter to invoke a zip program to compress the file

See Also

~~objects to See Also as help, ~~~

Examples

```
## Not run:
  data <- list("chr1"=data.frame(x=c(100,130,200),y=c(1.2,4.0,2.3)));
  writewig(data,"filename");

## End(Not run)
```

Index

add.broad.peak.regions, 6

densum, 7

find.binding.positions, 8, *19, 20*, *26*, *39*

get.binding.characteristics, 12, *38*
get.broad.enrichment.clusters, 13, *39*
get.conservative.fold.enrichment.profile, 14
get.conservative.fold.enrichment.profile2, 16
get.mser, 18, *20*
get.mser.interpolation, 19
get.smoothed.enrichment.mle, *17*, 21, *23*
get.smoothed.enrichment.mle2, 22
get.smoothed.tag.density, *15*, *21, 22*, 24

help, *40*

output.binding.results, 25

points_within, 26

read.arachne.tags, 28
read.bam.tags, 29
read.bin.maqmap.tags, 29
read.bowtie.tags, 30
read.eland.tags, 30
read.helicos.tags, 31
read.maqmap.tags, 33
read.meland.tags, 33
read.short.arachne.tags, 34
read.tagalign.tags, 35
remove.local.tag.anomalies, 37

select.informative.tags, *6*, *21*, *24*, 38
spp *(*spp-package*)*, 2
spp-package, 2

write.broadpeak.info, 39
write.narrowpeak.binding, 39
writewig, *15*, *22*, *25*, 40