Package ‘sequenza’

October 14, 2022

Title Copy Number Estimation from Tumor Genome Sequencing Data

Description Tools to analyze genomic sequencing data from paired normal-tumor samples, including cellularity and ploidy estimation; mutation and copy number (allele-specific and total copy number) detection, quantification and visualization.

Version 3.0.0

Date 2019-05-09

Depends R (>= 3.2.0)

Imports pbapply, squash, iotools, readr, seqminer, copynumber

Suggests testthat, knitr, rmarkdown, rmdformats

License GPL-3

URL `https://sequenzatools.bitbucket.io`, Mailing list:

`https://groups.google.com/forum/#!forum/sequenza-user-group`

BugReports `https://bitbucket.org/sequenzatools/sequenza/issues`

SystemRequirements pandoc (>= 1.12.3)

VignetteBuilder knitr, rmarkdown

Encoding UTF-8

NeedsCompilation no

Author Francesco Favero [aut, cre] (<`https://orcid.org/0000-0003-3684-2659`>),
Andrea Marion Marquard [rev] (<`https://orcid.org/0000-0003-2928-6017`>),
Tejal Joshi [rev] (<`https://orcid.org/0000-0002-0939-2982`>),
Aron Charles Eklund [aut, ths]
(<`https://orcid.org/0000-0003-0861-1001`>)

Maintainer Francesco Favero <`favero.francesco@gmail.com`>

Repository CRAN

Date/Publication 2019-05-09 13:50:04 UTC

R topics documented:

`baf.bayes` *Model allele-specific copy numbers with specified cellularity and ploidy parameters*

Description

Given a pair of cellularity and ploidy parameters, the function returns the most likely allele-specific copy numbers with the corresponding log-posterior probability of the fit, for given values of B-allele frequency and depth ratio.

Usage

```
baf.bayes(Bf, depth.ratio, cellularity, ploidy, avg.depth.ratio,
          sd.Bf = 0.1, sd.ratio = 0.5, weight.Bf = 1, weight.ratio = 1,
          CNt.min = 0, CNt.max = 7, CNn = 2,
          priors.table = data.frame(CN = CNt.min:CNt.max, value = 1),
          ratio.priority = FALSE)
mufreq.bayes(mufreq, depth.ratio, cellularity, ploidy, avg.depth.ratio,
       weight.mufreq = 100, weight.ratio = 100, CNt.min = 1, CNt.max = 7, CNn = 2,
          priors.table = data.frame(CN = CNt.min:CNt.max, value = 1))
```

Arguments

`Bf`	vector of B-allele frequencies (values can range from 0 to 0.5).
`mufreq`	vector of mutation frequencies (values can range from 0 to 1).
`depth.ratio`	vector of depth ratios.

`sd.ratio` standard deviation observed in the depth ratio measures in a segment

`sd.Bf` standard deviation observed in the B-allele frequency measures in a segment

`weight.Bf` vector of weights for B-allele frequency values.

`weight.mufreq` vector of weights for the mutation frequency values.

`weight.ratio` vector of weights for the depth ratio values.

`cellularity` fraction of tumor cells in the sample.

`ploidy` 2 * ratio between total DNA content in a tumor cell and a normal cell.

`avg.depth.ratio`
average normalized depth ratio.

`CNt.min` minimum copy number to consider in the model.

`CNt.max` maximum copy number to consider in the model.

`CNn` copy number of the normal genome.

`priors.table` data frame with columns `CN` and `value`, containing the copy numbers and the corresponding weights. To every copy number is assigned the value 1 as default, so any values different from 1 will change the corresponding weight.

`ratio.priority` logical, if TRUE only the depth ratio will be used to determine the copy number state, while the Bf value will be used to determine the number of B-alleles.

Details

`baf.bayes` and `mufreq.bayes` use a naive Bayesian approach to calculate the posterior probability of fitness of the data point with the model point resulting from the given values of cellularity and DNA-content.

Value

`CNt` copy number of the tumor cell at the tested point.

`A` number of A-alleles at the tested point.

`B` number of B-alleles at the tested point.

`CNn` copy number of the normal cell at the tested point (equal to CNn given as argument).

`Mt` number of mutated alleles at the tested point.

`LPP` log-posterior probability of model fitting at the given point/segment.

See Also

`baf.model.fit`, `mufreq.model.fit`.

Examples

```
  ## Not run:
data.file <-  system.file("extdata", "example.seqz.txt.gz", package = "sequenza")
# read all the chromosomes:
seqz.data  <- read.seqz(data.file)
# Gather genome wide GC-stats from raw file:
```

```
gc.stats <- gc.sample.stats(data.file)
gc.vect  <- setNames(gc.stats$raw.mean, gc.stats$gc.values)
# Read only one chromosome:
seqz.data  <- read.seqz(data.file, chr.name = 1)

# Correct the coverage of the loaded chromosome:
seqz.data$adjusted.ratio <- seqz.data$depth.ratio /
                           gc.vect[as.character(seqz.data$GC.percent)]
# Select the heterozygous positions
seqz.hom  <- seqz.data$zygosity.normal == 'hom'
seqz.het  <- seqz.data[!seqz.hom, ]
# Detect breakpoints
breaks <- find.breaks(seqz.het, gamma = 80, kmin = 10, baf.thres = c(0, 0.5))
# use heterozygous and homozygous position to measure segment values
seg.s1 <- segment.breaks(seqz.data, breaks = breaks)

# filter out small ambiguous segments, and conveniently weight the segments by size:
seg.filtered <- seg.s1[(seg.s1$end.pos - seg.s1$start.pos) > 10e6, ]
weights.seg  <- 150 + round((seg.filtered$end.pos -
                             seg.filtered$start.pos) / 1e6, 0)
# get the genome wide mean of the normalized depth ratio:
avg.depth.ratio <- mean(gc.stats$adj[,2])
# run the BAF model fit

CP <- baf.model.fit(Bf = seg.filtered$Bf, depth.ratio = seg.filtered$depth.ratio,
                    weight.ratio = weights.seg,
                    weight.Bf = weights.seg,
                    avg.depth.ratio = avg.depth.ratio,
                    cellularity = seq(0.1,1,0.01),
                    ploidy = seq(0.5,3,0.05))

confint <- get.ci(CP)
ploidy   <- confint$max.ploidy
cellularity <- confint$max.cellularity

#detect copy number alteration on the segments:

cn.alleles <- baf.bayes(Bf = seg.s1$Bf, depth.ratio = seg.s1$depth.ratio,
                        cellularity = cellularity, ploidy = ploidy,
                        avg.depth.ratio = 1)

head(cbind(seg.s1, cn.alleles))

# create mutation table:
mut.tab   <- mutation.table(seqz.data, mufreq.treshold = 0.15,
                            min.reads = 40, max.mut.types = 1,
                            min.type.freq = 0.9, segments = seg.s1)

mut.tab.clean <- na.exclude(mut.tab)

# Detect mutated alleles:
mut.alleles <- mufreq.bayes(mufreq = mut.tab.clean$F,
                            depth.ratio = mut.tab.clean$adjusted.ratio,
                          cellularity = cellularity, ploidy = ploidy,
                          avg.depth.ratio = avg.depth.ratio)
head(cbind(mut.tab.clean[,c("chromosome","position","F",
                          "adjusted.ratio", "mutation")],
          mut.alleles))


## End(Not run)
```

`baf.model.fit` *Model fitting using maximum a posteriori inference*

Description

Computes the log-posterior probability distribution for the specified range of cellularity and ploidy parameters

Usage

```
mufreq.model.fit(cellularity = seq(0.3, 1, by = 0.01),
    ploidy = seq(1, 7, by = 0.1), mc.cores = getOption("mc.cores", 2L),
    ...)
baf.model.fit(cellularity = seq(0.3, 1, by = 0.01),
    ploidy = seq(1, 7, by = 0.1), mc.cores = getOption("mc.cores", 2L),
    ...)
```

Arguments

`cellularity`	vector of cellularity values to be tested.
`ploidy`	vector of ploidy values to be tested.
`mc.cores`	number of cores to use, defined as in `pblapply`.
`...`	any argument accepted by `mufreq.bayes` or `baf.bayes`.

Details

`baf.model.fit` uses the function `baf.bayes` to infer the log-posterior probability of the model fit using the possible combinations of cellularity and ploidy values provided in the arguments. Similarly `mufreq.model.fit` fits the mutation/depth ratio model using the function `mufreq.bayes`. `baf.model.fit` is the defalt method used to infer cellularity and ploidy on segmented chromosomes. The `mufreq.model.fit` function estimates cellularity and ploidy using mutation frequency and depth ratio, however, the mutation data is more affected to background noise compared to the segmented B-allele frequency, hence it may give less accurate results.

Value

A list of three items:

`ploidy`	tested values of the ploidy parameter
`cellularity`	tested values of the cellularity parameter
`lpp`	log-posterior probability of each pair of cellularity/ploidy parameters.

See Also

`cp.plot` for visualization of the resulting object, and `get.ci` to extract confidence intervals.

Examples

```
## Not run:

data.file <-  system.file(”extdata”, ”example.seqz.txt.gz”,
    package = ”sequenza”)
# read all the chromosomes:
seqz.data <- read.seqz(data.file)
# Gather genome wide GC-stats from raw file:
gc.stats <- gc.sample.stats(data.file)
gc.normal.vect <- mean_gc(gc.stats$normal)
gc.tumor.vect <- mean_gc(gc.stats$tumor)
# Read only one chromosome:
seqz.data <- read.seqz(data.file, chr_name = ”1”)

# Correct the coverage of the loaded chromosome:
seqz.data$adjusted.ratio <- round((seqz.data$depth.tumor /
    gc.tumor.vect[as.character(seqz.data$GC.percent)]) /
    (seqz.data$depth.normal /
    gc.normal.vect[as.character(seqz.data$GC.percent)]), 3)
# Select the heterozygous positions
seqz.hom <- seqz.data$zygosity.normal == 'hom'
seqz.het <- seqz.data[!seqz.hom, ]
# Detect breakpoints
breaks <- find.breaks(seqz.het, gamma = 80, kmin = 10,
    baf.thres = c(0, 0.5))
# use heterozygous and homozygous position to measure segment values
seg.s1 <- segment.breaks(seqz.data, breaks = breaks)

# filter out small ambiguous segments, and conveniently weight
# the segments by size:
seg.filtered <- seg.s1[(seg.s1$end.pos - seg.s1$start.pos) > 3e6, ]
weights.seg  <- (seg.filtered$end.pos - seg.filtered$start.pos) / 1e6
# Set the average depth ratio to 1:
avg.depth.ratio <- 1
# run the BAF model fit
CP <- baf.model.fit(Bf = seg.filtered$Bf,
    depth.ratio = seg.filtered$depth.ratio, weight.ratio = weights.seg,
    weight.Bf = weights.seg, sd.ratio = seg.filtered$sd.ratio,
    sd.Bf = seg.filtered$sd.BAF, avg.depth.ratio = avg.depth.ratio,
        cellularity = seq(0.1, 1, 0.01), ploidy = seq(0.5, 3, 0.05))

    confint <- get.ci(CP)
    ploidy <- confint$max.ploidy
    cellularity <- confint$max.cellularity


## End(Not run)
```

---

chromosome.view *A graphical representation of multiple chromosomal features*

Description

A graphical representation of depth ratio, allele frequency and mutation frequency in multiple panels allineated by the coordinate of the same chromosome.

Usage

```
chromosome.view(baf.windows, ratio.windows, mut.tab = NULL,
    segments = NULL, min.N.baf = 1, min.N.ratio = 10000, main = "",
    vlines = FALSE, legend.inset = c(-20 * strwidth("a", units = "figure"),
    0), CNn = 2, cellularity = NULL, ploidy = NULL, avg.depth.ratio = NULL,
    model.lwd = 1, model.lty = "24", model.col = 1, x.chr.space = 10)
genome.view(seg.cn, info.type = "AB", ...)
```

Arguments

`baf.windows` matrix containing the windowed B-allele frequency values for one chromosome.

`ratio.windows` matrix containing the windowed depth ratio values for one chromosome.

`mut.tab` mutation table of one chromosome. If specified, the mutations will be drawn in a top panel. `mut.tab` must be output from the `mutation.table` function.

`segments` segmentation for one chromosome. If specified, the segmented B-allele frequency and depth ratio values will be shown as red lines.

`min.N.baf` minimum number of observations required in a BAF window for plotting.

`min.N.ratio` minimum number of observations required in a depth ratio window for plotting.

`CNn` copy number of the germline genome.

`vlines` logical, if TRUE the plot will include dotted vertical lines corresponding to segment breaks.

`cellularity` fraction of tumor cells in the sample.

`ploidy` value of the estimated `ploidy` parameter.

`avg.depth.ratio`
the average value of the normalized depth ratio.

`main` main title of the plot.

`legend.inset` the inset argument to pass to the `legend` function. Defines the distance between the mutation legend and the plot border.

`model.lwd` width of the theoretical lines, if the segments matrix contains the columns A, B and CNt.

`model.lty` line type of the theoretical lines, if the segments matrix contains the columns A, B and CNt.

`model.col` color of the theoretical lines, if the segments matrix contains the columns A, B and CNt.

`x.chr.space` step in megabase on the positions to visualize on the x-axis.

`seg.cn` genome wide segments, with the columns A, B and CNt.

`info.type` information to plot in `genome.view`. Available options are "CNt" for total copy numbers and "AB" (default) for the alleles specific copy number.

`...` optional arguments passed to `plot`.

Details

`chromosome.view` is a plotting function based on the default `plot` function and `par` to display multiple panels. The plotting function `plotWindows` is used to plot the binned data of `depth-ratio` and `b-allele` frequency. The function displays the observations reulting from the sequencing post-procssing as well the results of the model.

See Also

`windowValues`, `find.breaks`.

Examples

```
## Not run:

data.file <-  system.file("extdata", "example.seqz.txt.gz",
    package = "sequenza")
# read all the chromosomes:
seqz.data  <- read.seqz(data.file)
# Gather genome wide GC-stats from raw file:
gc.stats <- gc.sample.stats(data.file)
gc.vect  <- setNames(gc.stats$raw.mean, gc.stats$gc.values)
# Read only one chromosome:
seqz.data  <- read.seqz(data.file, chr.name = 1)

# Correct the coverage of the loaded chromosome:
seqz.data$adjusted.ratio <- seqz.data$depth.ratio /
    gc.vect[as.character(seqz.data$GC.percent)]
# Select the heterozygous positions
seqz.hom  <- seqz.data$zygosity.normal == 'hom'
seqz.het  <- seqz.data[!seqz.hom, ]
# Detect breakpoints
breaks <- find.breaks(seqz.het, gamma = 80, kmin = 10, baf.thres = c(0, 0.5))
# use heterozygous and homozygous position to measure segment values
seg.s1 <- segment.breaks(seqz.data, breaks = breaks)
```

```
    # Binning the values of depth ratio and B allele frequency
    seqz.r.win <- windowValues(x = seqz.data$adjusted.ratio,
        positions = seqz.data$position, chromosomes = seqz.data$chromosome,
        window = 1e6, overlap = 1, weight = seqz.data$depth.normal)

    seqz.b.win <- windowValues(x = seqz.het$Bf,
        positions = seqz.het$position, chromosomes = seqz.het$chromosome,
        window = 1e6, overlap = 1, weight = round(x = seqz.het$good.reads,
            digits = 0))
    # create mutation table:
    mut.tab   <- mutation.table(seqz.data, mufreq.treshold = 0.15,
        min.reads = 40, max.mut.types = 1, min.type.freq = 0.9,
        segments = seg.s1)
    # chromosome view without parametes:
    chromosome.view(mut.tab = mut.tab[mut.tab$chromosome == ”1”,],
        baf.windows = seqz.b.win[[1]], ratio.windows = seqz.r.win[[1]],
        min.N.ratio = 1, segments = seg.s1[seg.s1$chromosome == ”1”,],
        main = ”Chromosome 1”)

    # filter out small ambiguous segments, and weight the segments by size:
    seg.filtered <- seg.s1[(seg.s1$end.pos - seg.s1$start.pos) > 10e6, ]
    weights.seg  <- 150 + round((seg.filtered$end.pos -
                             seg.filtered$start.pos) / 1e6, 0)
    # get the genome wide mean of the normalized depth ratio:
    avg.depth.ratio <- mean(gc.stats$adj[,2])
    # run the BAF model fit

    CP <- baf.model.fit(Bf = seg.filtered$Bf, depth.ratio = seg.filtered$depth.ratio,
        weight.ratio = weights.seg, weight.Bf = weights.seg,
        avg.depth.ratio = avg.depth.ratio, cellularity = seq(0.1,1,0.01),
        ploidy = seq(0.5,3,0.05))

    confint <- get.ci(CP)
    ploidy   <- confint$max.ploidy
    cellularity <- confint$max.cellularity
    #detect copy number alteration on the segments:
    cn.alleles <- baf.bayes(Bf = seg.s1$Bf, depth.ratio = seg.s1$depth.ratio,
        cellularity = cellularity, ploidy = ploidy, avg.depth.ratio = 1)

    seg.s1 <- cbind(seg.s1, cn.alleles)

    # Chromosome view with estimated paramenters:
    chromosome.view(mut.tab = mut.tab[mut.tab$chromosome == ”1”,],
        baf.windows = seqz.b.win[[1]], ratio.windows = seqz.r.win[[1]],
        min.N.ratio = 1, segments = seg.s1[seg.s1$chromosome == ”1”,],
        main = ”Chromosome 1”, cellularity = cellularity, ploidy = ploidy,
        avg.depth.ratio = 1, BAF.style = ”lines”)

## End(Not run)
```

`CP.example` *Example of cellularity and ploidy results*

Description

Examples of results from the maximum a posteriori estimation from a set of cellularity and ploidy values, as returned by the functions `baf.model.fit` and `mufreq.model.fit`.

Usage

```
data(CP.example)
```

Format

A list containing three items:

ploidy numeric vector of tested ploidy values.

cellularity numeric vector of tested cellularity values.

lpp numeric matrix of log-posterior probability for each *(ploidy, cellularity)* pair.

Examples

```
data(CP.example)
str(CP.example)

## Visualization of the object
image(x = CP.example$ploidy,
    y = CP.example$cellularity,
    z = CP.example$lpp)

## A better plot
cp.plot(CP.example)
cp.plot.contours(CP.example, add = TRUE)
```

`cp.plot` *Plot log-posterior probability for the output of the* `sequenza.fit` *function*

Description

This function uses the `colorgram` function from the package **squash** to plot log-posterior probability for the tested combinations of cellularity and ploidy

Usage

```
cp.plot(cp.table, xlab = "Ploidy", ylab = "Cellularity",
        zlab = "Scaled rank LPP",
        colFn = colorRampPalette(c('white', 'lightblue')), ...)
cp.plot.contours(cp.table, likThresh = c(0.95), alternative = TRUE,
                 col = palette(), legend.pos = "bottomright", pch = 18,
                 alt.pch = 3, ...)
get.ci(cp.table, level = 0.95)
```

Arguments

`cp.table` list, as output from `baf.model.fit` or `mufreq.model.fit`.

`xlab` xlab parameter as in the function `colorgram`.

`ylab` ylab parameter as in the function `colorgram`.

`zlab` zlab parameter as in the function `colorgram`.

`colFn` colFn parameter as in the function `colorgram`.

`likThresh` vector of quantiles to define tresholds for the confindent regions.

`alternative` boolean parameter, if `TRUE` the alternative solutions are computed and plotted.

`col` vector of colors.

`legend.pos` position for placing the legend.

`pch` character used to indicate the point estimate.

`alt.pch` if `alternative` is set to `TRUE` defines the character to indicate alternative solutions.

`...` additional arguments accepted by the function `colorgram` for `cp.plot`, or `contour` for `cp.plot.contours`.

`level` decimal value of the confidence interval

Value

The `get.ci` function returns a list with 6 items:

`values.ploidy` matrix of ploidy values with respective posterior probability.

`confint.ploidy` boundaries of the confidence interval of the estimated ploidy.

`max.ploidy` point estimate of the ploidy value that has the maximum posterior probability.

`values.cellularity` matrix of cellularity values with respective posterior probability.

`confint.cellularity` boundaries of the confidence interval of the estimated cellularity.

`max.cellularity` point estimate of the cellularity value that has the maximum posterior probability.

Examples

```
data(CP.example)
cp.plot(CP.example)
cp.plot.contours(CP.example, add = TRUE)

# Plot more contours
cp.plot(CP.example)
cp.plot.contours(CP.example, likThresh = c(0.95, 0.9999), add = TRUE)

# Return the 95% confidence interval
CP.example.ci <- get.ci(CP.example)
str(CP.example.ci)
```

example.seqz *Example “seqz” data*

Description

The “seqz” file is produced by sequenza-utils and typically has the file extension ‘.seqz’. The data here is representative of a seqz file derived from an exome-sequenced tumor sample, such as could be obtained from TCGA.

Usage

```
data(example.seqz)
```

Format

A data frame with 53937 rows and 14 columns:

[,1]	chromosome	Chromosome name
[,2]	position	Base position
[,3]	base.ref	Base in the reference genome
[,4]	depth.normal	Read depth in the normal sample
[,5]	depth.tumor	Read depth in the tumor sample
[,6]	depth.ratio	Ratio of depth.tumor and depth.normal
[,7]	Af	A-allele frequency in the tumor sample
[,8]	Bf	B-allele frequency in the tumor sample, in heterozygous positions only
[,9]	zygosity.normal	Zygosity of the normal sample: "hom" for homozygous or "het" for heterozygous
[,10]	GC.percent	% GC content
[,11]	good.reads	Number of reads from the tumor sample which pass the quality threshold
[,12]	AB.normal	Base(s) found in the normal sample, sorted by allele frequency if more than one
[,13]	AB.tumor	Base(s) found in the tumor sample *but not* in the normal specimen, with their observed frequencies,
[,14]	tumor.strand	Identical to AB.tumor but indicating, for each variant base, the fraction of reads oriented in the forw

Details

example.seqz can be loaded in the standard R way via data(example.seqz), or it can be read from a text file using read.seqz. The former is useful for examples and testing, whereas the latter is representative of the standard workflow.

Source

This is derived from a TCGA specimen, but has been scrambled to anonymize the source. The reference genome is hg19. The GC content was calculated in 50-base windows.

find.breaks *Segmentation of sequencing data using an allele-specific copy number algorithm*

Description

This function uses aspcf or pcf from the package **copynumber** to segment depth ratio and B-allele frequency obtained from sequencing data.

Usage

```
find.breaks(seqz.baf, gamma = 80, kmin = 10, baf.thres = c(0, 0.5),
            verbose = FALSE, seg.algo = "aspcf", ...)
segment.breaks(seqz.tab, breaks, min.reads.baf = 1, weighted.mean = TRUE)
```

Arguments

seqz.baf	an seqz file containing only the heterozygous positions.
seqz.tab	a complete seqz file.
gamma, kmin, baf.thres, verbose	arguments passed to the segmentation algorithm.
breaks	breaks as output by find.breaks.
min.reads.baf	threshold on the depth of the positions included to calculate the average BAF for segment.
weighted.mean	boolean to select if the segments have to calculated using the read depth as a weights to calculate depth ratio and B-allele frequency means.
seg.algo	Selects the algorithm used for the segmentation. Available options are aspcf of pcf.
...	additional arguments passed to aspcf.

Details

copynumber is a package to perform efficient segmentation of SNP-array data. The function find.breaks uses the algorithms from the **copynumber** package to find break points, where the default parameters have been optimized for sequencing data, but a careful choice of an optimal gamma value is advised.

Examples

```
  ## Not run:

data.file <-  system.file("extdata", "example.seqz.txt.gz", package = "sequenza")
# read all the chromosomes:
seqz.data  <- read.seqz(data.file)
# Gather genome wide GC-stats from raw file:
gc.stats <- gc.sample.stats(data.file)
gc.vect  <- setNames(gc.stats$raw.mean, gc.stats$gc.values)
# Read only one chromosome:
seqz.data  <- read.seqz(data.file, chr.name = 12)

# Correct the coverage of the loaded chromosome:
seqz.data$adjusted.ratio <- seqz.data$depth.ratio /
                           gc.vect[as.character(seqz.data$GC.percent)]
# Select the heterozygous positions
seqz.hom  <- seqz.data$zygosity.normal == 'hom'
seqz.het  <- seqz.data[!seqz.hom, ]
# Detect breakpoints
breaks <- find.breaks(seqz.het, gamma = 80, kmin = 10, baf.thres = c(0, 0.5))
# use heterozygous and homozygous position to measure segment values
segment.breaks(seqz.data, breaks = breaks)

## End(Not run)
```

gc.sample.stats *Collect display and correct GC-content related coverage bias*

Description

Collect information and perform statistics of depth of coverage in relation with GC-content.

Usage

```
gc.sample.stats(file, col_types = "c--dd----d----", buffer = 33554432,
                parallel = 2L, verbose = TRUE)
gc.summary.plot(gc_list, mean.col = 1, median.col = 2,
   scale.subset = 1.5, ...)
mean_gc(gc_list)
median_gc(gc_list)
```

Arguments

`file` name of a file in the seqz format.

`col_types` a string describing the classes of each columns of the input file (see `read_tsv`). The default value corresponds to the columns of a seqz file used for carculating GC statistics.

`buffer` maximal size of each chunk in bytes(see `chunk.apply`).

`parallel` integer, number of threads used to process a seqz file (see `chunk.apply`).

`verbose` logical. If TRUE (the default) the function retuns information in the console.

`gc_list` a normal or tumor list resulting from the `gc.sample.stats` function.

`mean.col` color for the mean in the summary plot.

`median.col` color for the median in the summary plot.

`scale.subset` scale the depth values to sho in the plot. A value of 1 will show the average depth at the center of the plot.

`...` additional parametrers from `colorgram`.

Details

`gc.sample.stats` extracts depths and GC-content inforation for the tumor and the control samples from an seqz file it returns a list with 3 elements: `file.metrics`, `normal` and `tumor`.

`file.metrics` is a `data.frame` serving as index of the seqz file; the `normal` and `tumor` objects contains each 3 ojects: `gc`, `depth` and `n`.

`gc` and `depth` are vectors containing the recorded values of, respectively, GC and coverage depth. the `n` object is a matrix `gcxdepth`, recording the number of time a certain `gc`/`depth` pairs is observed in the data.

Value

A list with the following elements:

`file.metrics` index of the seqz file.

`tumor` GC and coverage depth observations in the tumor sample.

`normal` GC and coverage depth observations in the control sample.

Examples

```
## Not run:

data.file <-  system.file("extdata", "example.seqz.txt.gz", package = "sequenza")
# read all the chromosomes:
gc_info <- gc.sample.stats(data.file)

# mean values of depth coverage vs GC content

mean_gc(gc_info$normal)

# plot the information for the tumor and normal samples
par(mfrow=c(1, 2))
gc.summary.plot(gc_info$normal, main = "Normal GC stats")
gc.summary.plot(gc_info$tumor, main = "Tumor GC stats")

## End(Not run)
```

model.points *Generate B-allele frequency, mutation frequency and depth ratios at given model points, cellularity and ploidy values*

Description

The baf.model.points and mufreq.model.points functions combine theoretical_baf, theoretical_mufreq and theoretical_depth_ratio to model the theoretical respective values at known values of cellularity and ploidy.

Usage

```
baf.model.points(cellularity, ploidy, baf_types, avg.depth.ratio)
mufreq.model.points(cellularity, ploidy, mufreq_types, avg.depth.ratio)
```

Arguments

`cellularity` fraction of tumor cells in the sample.

`ploidy` 2 * ratio between total DNA content in a tumor cell and a normal cell.

`baf_types` matrix with the sets of copy numbers and number of mutated alleles over which to model mutation frequency and depth ratio. The matrix can be generated with `baf.types.matrix`.

`mufreq_types` matrix with the sets of copy numbers and number of mutated alleles over which to model mutation frequency and depth ratio. The matrix can be generated with `mufreq.types.matrix`.

`avg.depth.ratio`
average normalized depth ratio.

Details

The baf.model.points and mufreq.model.points functions generate the theoretical values of B-allele frequency, mutation frequency and depth ratio for the given type tags. To learn more about type tags see `types.matrix`.

Value

For baf.model.points a data.frame with two columns:

`BAF` modelled values of B-allele frequency.

`depth_ratio` modelled values of depth ratio.

For mufreq.model.points a data.frame with two columns:

`mufreqs` modelled values of mutation frequency.

`depth_ratio` modelled values of depth ratio.

See Also

types.matrix, theoretical.depth.ratio, theoretical.baf theoretical.mufreq.

Examples

```
# Simulate a cellularity of 0.5, ploidy of 2 and types from min CNt 0
# and max = 4 on an originally diploid genome:
types <- baf.types.matrix(CNt.min = 0, CNt.max = 4, CNn = 2)
cbind(types, baf.model.points(cellularity = 0.5, ploidy = 2,
    baf_types = types, avg.depth.ratio = 1))
# Simulate a cellularity of 0.5, ploidy of 2 and types from min CNt 0
# and max = 4 on an originally monoallelic genome:
types <- mufreq.types.matrix(CNt.min = 0, CNt.max = 4, CNn = 1)
cbind(types, mufreq.model.points(cellularity = 0.5, ploidy = 2,
    mufreq_types = types, avg.depth.ratio = 1))
```

---

mutation.table *Identify mutations*

Description

This function extracts positions from an seqz file that differ from the normal genome, applying various filters.

Usage

```
mutation.table(seqz.tab, mufreq.treshold = 0.15, min.reads = 40, min.reads.normal = 10,
         max.mut.types = 3, min.type.freq = 0.9, min.fw.freq = 0, segments = NULL)
```

Arguments

`seqz.tab` an seqz table, as output from read.seqz.

`mufreq.treshold` mutation frequency threshold.

`min.reads` minimum number of reads above the quality threshold to accept the mutation call.

`min.reads.normal` minimum number of reads used to determine the genotype in the normal sample.

`max.mut.types` maximum number of different base substitutions per position. Integer from 1 to 3 (since there are only 4 different bases). Default is 3, to accept “noisy” mutation calls.

`min.type.freq` minimum frequency of aberrant types.

`min.fw.freq` minimum frequency of variant reads detected in the forward strand. Setting it to 0, all the variant calls with strand frequency in the interval outside 0 and 1, margin not comprised, would be discarded.

`segments` if specified, the values of depth ratio would be taken from the segments rather than from the raw data.

Details

Calling mutations in impure tumor samples is a difficult task, because the degree of contamination by normal cells affects the measured mutation frequency. In highly impure samples, where the normal cells comprise the major component of the sample, mutations can be so diluted that it can be difficult to distinguish them from sequencing errors.

The function `mutation.table` tries to separate true mutations from sequencing errors, based on the given threshold. In samples with low contamination, it should even be possible to catch sub-clonal mutations using this function.

This function identifes only those mutations occuring in positions that are homozygous in the normal genome.

Value

A data frame, which in addition to some of the columns of the seqz table, contains the following two columns:

`F`	the mutation frequency
`mutation`	a character representation of the mutation. For example, a mutation from '`A`' in the normal to '`G`' in the tumor is annotated as '`A>G`'.

Examples

```
   ## Not run:

data.file <-  system.file("extdata", "example.seqz.txt.gz", package = "sequenza")
seqz.data  <- read.seqz(data.file)

## Normalize coverage by GC-content
gc.stats <- gc.norm(x = seqz.data$depth.ratio,
                    gc = seqz.data$GC.percent)
gc.vect  <- setNames(gc.stats$raw.mean, gc.stats$gc.values)
seqz.data$adjusted.ratio <- seqz.data$depth.ratio /
                           gc.vect[as.character(seqz.data$GC.percent)]

## Extract mutations
mut.tab   <- mutation.table(seqz.data, mufreq.treshold = 0.15,
                            min.reads = 40, max.mut.types = 1,
                            min.type.freq = 0.9)
mut.tab <- na.exclude(mut.tab)

## End(Not run)
```

plotWindows *Plot a binned values of a chromosome*

Description

The plotWindows function visualizes a data.frame produced by the windowValues or windowBf functions.

Usage

```
plotWindows(seqz.window, m.lty = 1, m.lwd = 3, m.col = "black",
    q.bg = "lightblue", log2.plot = FALSE, n.min = 1, xlim, ylim,
    add = FALSE, ...)
```

Arguments

seqz.window	data frame of base-pair windows and corresponding quartiles to be plotted. A list of such data frames can be output from windowValues or windowBf.
m.lty	line type used for plotting mean values.
m.lwd	line width used for plotting mean values.
m.col	line color used for plotting mean values.
q.bg	background color for the area between the 0.25 and 0.75 quartiles.
log2.plot	logical, if TRUE values are log2 scaled.
n.min	minimum number of data points required for a binned window to be plotted.
xlim	limits of the x axis.
ylim	limits of the y axis.
add	logical, if TRUE the plot will be added to an existing opened device.
...	any other arguments accepted by plot.

See Also

chromosome.view,

Examples

```
data.file <-  system.file("extdata", "example.seqz.txt.gz",
    package = "sequenza")
seqz.data <- read.seqz(data.file)
# 1Mb windows, each window is overlapping with 1 other adjacent
# window: depth ratio
seqz.ratio <- windowValues(x = seqz.data$depth.ratio,
    positions = seqz.data$position, chromosomes = seqz.data$chromosome,
    window = 1e6, weight = seqz.data$depth.normal, start.coord = 1,
    overlap = 1)
```

```
plotWindows(seqz.ratio[[1]], log2.plot = FALSE, ylab = "Depth ratio",
    xlab = "Position (bases)", main = names(seqz.ratio)[1], las = 1,
    n.min = 1, ylim = c(0, 2.5))

plotWindows(seqz.ratio[[17]], log2.plot = FALSE, ylab = "Depth ratio",
    xlab = "Position (bases)", main = names(seqz.ratio)[1], las = 1,
    n.min = 1, ylim = c(0, 2.5))
```

`read.seqz` *Read a seqz or acgt format file*

Description

Efficiently reads a seqz file into R.

Usage

```
read.seqz(file, n_lines = NULL, col_types = "ciciidddcddccc", chr_name = NULL,
     buffer = 33554432, parallel = 1,
     col_names = c("chromosome", "position", "base.ref", "depth.normal",
         "depth.tumor", "depth.ratio", "Af", "Bf", "zygosity.normal",
         "GC.percent", "good.reads", "AB.normal", "AB.tumor",
         "tumor.strand"),...)
```

Arguments

`file` file name

`col_types` a string describing the classes of each columns of the input file (see `read_tsv`). The default value corresponds to the columns of a seqz file.

`chr_name` if specified, only the selected chromosome will be extracted instead of the entire file. For `tabix`-indexed files this argument can also be used to extract coordinated-selected genomic regions. E.g. `chr_name="5:1-1000000"` will select the first megabase of chromosome 5.

`n_lines` vector of length 2 specifying the first and last line to read from the file. If specified, only the selected portion of the file will be used.

`buffer` maximal size of each chunk in bytes(see `chunk.apply`).

`parallel` integer, number of threads used to process a seqz file (see `chunk.apply`).

`col_names` names of the columns of the seqz file. The default corresponds to the column names of a seqz file.

`...` any arguments accepted by `read_tsv`.

Format

A seqz file is a tab-separated text file with 14 columns and a header row. The first 3 columns are derived from the original `pileup` file and contain:

chromosome the chromosome name

position the base position

base.ref the base in the reference genome. Note that this is NOT necessarily the same base as in the normal specimen. The remaining 10 columns contain the following information:

depth.normal read depth observed in the normal sample

depth.tumor read depth observed in the tumor sample

depth.ratio ratio of `depth.tumor` and `depth.normal`

Af A-allele frequency observed in the tumor sample

Bf B-allele frequency observed in the tumor sample in heterozygous positions

zygosity.normal zygosity of the reference sample. "hom" corresponds to AA or BB, whereas "het" corresponds to AB or BA

GC.percent GC-content (percent), calculated from the reference genome in fixed nucleotide windows

good.reads number of reads that passed the quality threshold (threshold specified in the pre-processing software), in the tumor specimen

AB.normal base(s) found in the germline sample; for heterozygous positions AB are sorted using the values of Af and Bf respectively

AB.tumor base(s) found in the tumor sample not present in the normal specimen. The field include all the variants found in the tumor alignment, separated by a colon. Each variant contains the base and the observed frequency

tumor.strand frequency of the variant nucleotides detected on the forward orientation. The field have a consistent structure with `AB.tumor`, indicating the fraction, relative to the total number of reads presenting the specific variant, orientated in the forward direction

Details

`read.seqz` is a function that allows to efficiently access a `seqz` file by chromosome or by line numbers. The function can also access coordinate specific regions with `tabix`-indexed `seqz` files. The specific content of a `seqz` file is explained in the `value` section.

See Also

`read_delim`.

Examples

```
## Not run:

 data_file <-  system.file("extdata", "example.seqz.txt.gz", package = "sequenza")

 ## read chromosome 1 from an seqz file.
```

```
    seqz_data <- read.seqz(data_file, chr_name = 1)

    ## Fast access to chromosome X using the file metrics
    gc.stats <- gc.sample.stats(data_file)
    chrX <- gc.stats$file.metrics[gc.stats$file.metrics$chr == "X", ]
    seqz.data <- read.seqz(data_file, n_lines = c(chrX$start, chrX$end))

    ## Compare the running time of the two different methods.
    system.time(seqz.data <- read.seqz(data_file, n_lines = c(chrX$start, chrX$end)))
    system.time(seqz.data <- read.seqz(data_file, chr_name = "X"))


## End(Not run)
```

sequenza *Sequenza convenience functions for standard analysis*

Description

These three functions are intended to be the main user interface of the package, to run several of the functions of sequenza in a standardized pipeline.

Usage

```
sequenza.extract(file, window = 1e6, overlap = 1,
  gamma = 80, kmin = 10, gamma.pcf = 140, kmin.pcf = 40,
  mufreq.treshold = 0.10, min.reads = 40, min.reads.normal = 10,
  min.reads.baf = 1, max.mut.types = 1, min.type.freq = 0.9,
  min.fw.freq = 0, verbose = TRUE, chromosome.list = NULL,
  breaks = NULL, breaks.method = "het", assembly = "hg19",
  weighted.mean = TRUE, normalization.method = "mean",
  ignore.normal = FALSE, parallel = 1, gc.stats = NULL,
  segments.samples = FALSE)

sequenza.fit(sequenza.extract, female = TRUE, N.ratio.filter = 10,
             N.BAF.filter = 1, segment.filter = 3e6,
             mufreq.treshold = 0.10, XY = c(X = "X", Y = "Y"),
             cellularity = seq(0.1,1,0.01), ploidy = seq(1, 7, 0.1),
             ratio.priority = FALSE, method = "baf",
             priors.table = data.frame(CN = 2, value = 2),
             chromosome.list = 1:24, mc.cores = getOption("mc.cores", 2L))

sequenza.results(sequenza.extract, cp.table = NULL, sample.id, out.dir = getwd(),
                cellularity = NULL, ploidy = NULL, female = TRUE, CNt.max = 20,
                  ratio.priority = FALSE, XY = c(X = "X", Y = "Y"),
                  chromosome.list = 1:24)
```

Arguments

`file` the name of the seqz file to read.

`window` size of windows used when plotting mean and quartile ranges of depth ratios and B-allele frequencies. Smaller windows will take more time to compute.

`overlap` integer specifying the number of overlapping windows.

`gamma, kmin` arguments passed to `aspcf` from the **copynumber** package.

`gamma.pcf, kmin.pcf`
arguments passed to `pcf` from the **copynumber** package. The arguments are effective only when `breaks.method` is set to "full".

`mufreq.treshold`
mutation frequency threshold.

`min.reads` minimum number of reads above the quality threshold to accept the mutation call.

`min.reads.normal`
minimum number of reads used to determine the genotype in the normal sample.

`min.reads.baf` threshold on the depth of the positions included to calculate the average BAF for segment.

`max.mut.types` maximum number of different base substitutions per position. Integer from 1 to 3 (since there are only 4 bases). Default is 3, to accept "noisy" mutation calls.

`min.type.freq` minimum frequency of aberrant types.

`min.fw.freq` minimum frequency of variant reads detected in the forward strand. Setting it to 0, all the variant calls with strand frequency in the interval outside 0 and 1, margin not comprised, would be discarded.

`verbose` logical, indicating whether to print information about the chromosome being processed.

`chromosome.list`
vector containing the index or the names of the chromosome to include in the model fitting.

`breaks` Optional data.frame in the format chrom, start.pos, end.pos, defining a pre-existing segmentation. When the argument is set the built-in segmentation will be skipped in favor of the suggested breaks.

`breaks.method` Argument indicating the resolution of the segmentation. Possible values are `fast`, `het` and `full`, where `fast` allows the lower resolution and `full` the higher. Custom values of `gamma` and `kmin` need to be adjusted to have optimal results.

`assembly` assembly version of the genome, see `aspcf` or `pcf`.

`weighted.mean` boolean to select if the segments should be calculated using the read depth as weights to calculate depth ratio and B-allele frequency means.

`normalization.method`
string defining the operation to perform during the GC-normalization process. Possible values are `mean` (default) and `median`. A `median` normalization is preferable with noisy data.

`ignore.normal` boolean, when set to TRUE the process will ignore the normal coverage and perform the analysis by using the normalized tumor coverage.

`parallel` integer, number of threads used to process a seqz file (see `chunk.apply`).

`gc.stats` object returned from the function `gc.sample.stats`. If `NULL` the object will be computed from the input file.

`segments.samples`
EXPERIMENTAL. Segment both tumor and normal samples separately, and add it to the QC plots.

`sequenza.extract`
a list of objects as output from the `sequenza.extract` function.

`method` method to use to fit the data; possible values are `baf` to use `baf.model.fit` or `mufreq` to use the `mufreq.model.fit` function to fit the data.

`cp.table` a list of objects as output from the `sequenza.fit` function.

`female` logical, indicating whether the sample is male or female, to properly handle the X and Y chromosomes. Implementation only works for the human normal karyotype.

`CNt.max` maximum copy number to consider in the model.

`N.ratio.filter` threshold of minimum number of observation of depth ratio in a segment.

`N.BAF.filter` threshold of minimum number of observation of B-allele frequency in a segment.

`segment.filter` threshold segment length (in base pairs) to filter out short segments, that can cause noise when fitting the cellularity and ploidy parameters. The threshold will not affect the allele-specific segmentation.

`XY` character vector of length 2 specifying the labels used for the X and Y chromosomes.

`cellularity` vector of candidate cellularity parameters.

`ploidy` vector candidate ploidy parameters.

`priors.table` data frame with the columns `CN` and `value`, containing the copy numbers and the corresponding weights. To every copy number is assigned the value 1 as default, so every values different then 1 will change the corresponding weight.

`ratio.priority` logical, if TRUE only the depth ratio will be used to determine the copy number state, while the Bf value will be used to determine the number of B-alleles.

`sample.id` identifier of the sample, to be used as a prefix for saved objects.

`out.dir` output directory where the files and objects will be saved.

`mc.cores` legacy argument to set the number of cores, but it refers to the `cl` of `pblapply`. It uses `mclapply` when set to an integer.

Details

The first function, `sequenza.extract`, utilizes a range of functions from the sequenza package to read the raw data, normalize the depth.ratio for GC-content bias, perform allele-specific segmentation, filter for noisy mutations and bin the raw data for plotting. The computed objects are returned as a single list object.

The segmentation by default is performed using only the heterozygous position and the aspcf function from **copynumber** package. The full option in the breaks.method argument allow to combine results of the segmentation of all the data available, using the pcf function, and the default aspcf using only the heterozygous positions.

The second function, sequenza.fit, accepts the output from sequenza.extract and calls baf.model.fit to calculate the log-posterior probability for all pairs of the candidate ploidy and cellularity parameters.

The third function, sequenza.results, saves a number of objects in a specified directory (default is the working directory). The objects are:

- The list of segments with resulting copy numbers and major and minor alleles.
- The candidate mutation list with variant allele frequency, and copy number and number of mutated allele, in relation of the clonal population (for sub-clonal population it needs to be processed with further methods).
- A plot of all the chromosomes in one image, representing the major and minor alleles and the absolute copy number changes (genome_view).
- Multiple plots with one chromosome per image, representing copy-number, B-allele frequency and mutation in parallel (chromosome_view).
- Results of the model fitting (CP_contours and confints_CP)
- A summary of the copy number state of the sample (CN_bars).

See Also

genome.view, baf.bayes, cp.plot, get.ci.

Examples

```
  ## Not run:

data.file <-  system.file("extdata", "example.seqz.txt.gz",
                package = "sequenza")
test <- sequenza.extract(data.file)
test.CP   <- sequenza.fit(test)
sequenza.results(test, test.CP, out.dir = "example",
                 sample.id = "example")


## End(Not run)
```

theoretical.baf *Calculates cellularity and ploidy dependent model points*

Description

Calculates the theoretically expected values of BAF, mutation frequency or depth ratio for given values of cellularity, ploidy and copy number.

Usage

```
theoretical.depth.ratio(CNt, cellularity, ploidy, CNn = 2,
    normal.ploidy = 2, avg.depth.ratio = 1)
theoretical.baf(CNt, B, cellularity, CNn = 2)
theoretical.mufreq(CNt, Mt, cellularity, CNn = 2)
```

Arguments

`CNn` copy number in the normal sample.

`CNt` copy number in the tumor sample.

`B` number of B-alleles in the tumor sample.

`Mt` number of alleles carrying a mutation in the tumor sample.

`cellularity` fraction of tumor cells in the sample.

`ploidy` 2 * ratio between total DNA content in a tumor cell and a normal cell.

`normal.ploidy` ploidy value in the normal sample. Default is 2 for a diploid cell.

`avg.depth.ratio` average normalized depth ratio.

Details

The observed B-allele frequency, depth ratio and mutation frequency are affected by the cellularity of the tumor sample, which is the inverse of the degree of contamination by normal cells. Three functions are included, which for know values of cellularity and ploidy they produce the expected values of B-allele frequency, mutation frequency or depth ratio.

`theoretical.baf` returns a dataframe with the possible copy numbers of A and B alleles, along with their corresponding B-allele frequency and the total copy number state (always the sum of A+B).

`theoretical.depth.ratio` returns the theoretical depth ratio at a single specific position, given values of cellularity, ploidy, the ratio between the tumor copy number and the normal copy number at that position, and the average depth ratio of the sample.

`theoretical.mufreq` returns the theoretical mutation frequency at a single specific position, given values of cellularity, copy number in the normal and tumor samples at that position, and the number of mutated alleles.

See Also

`model.points`

types.matrix *Creates a matrix of type tags*

Description

Type tags are a utensil to distinguish genomic positions by their copy number state, number A and B alleles and the number of mutated alleles. This function creates a matrix of all possible type tags, given the copy number of the normal sample and the range of possible copy numbers in the tumor sample.

Usage

```
baf.types.matrix(CNt.min, CNt.max, CNn = 2)
mufreq.types.matrix(CNt.min, CNt.max, CNn = 2)
```

Arguments

`CNt.min`	minimum copy number in the tumor.
`CNt.max`	maximum copy number in the tumor.
`CNn`	copy number of the normal sample.

Details

A type consists of 3 integers signifying the copy number in the normal and tumor samples and the number of B alleles (`baf.types.matrix`) or mutated alleles (`mufreq.types.matrix`). The two functions return all the possible types combination within the range of tumor copy numbers in the arguments (`CNt.min:CNt.max`).

Value

`baf.types.matrix` returns a data.frame with the 3 columns:

`CNn`	number of alleles in the normal sample.
`CNt`	numbers of alleles in the tumor sample.
`B`	number of B alleles in the tumor sample.

`mufreq.types.matrix` returns a data.frame with the 3 columns:

`CNn`	number of alleles in the normal sample.
`CNt`	numbers of alleles in the tumor sample.
`Mt`	number of mutated alleles in the tumor sample.

See Also

`theoretical_mufreq`, `theoretical_depth_ratio`, `theoretical_baf`, `model_points`.

Examples

```
## Generate matrix types from 0 to 4 copy number, being the
## non-tumor chromosome diploid.
baf.types.matrix(CNt.min = 0, CNt.max = 4, CNn = 2 )

## Generate matrix types from 0 to 4 copy number, being the
## non-tumor chromosome monoploid.
mufreq.types.matrix(CNt.min = 0, CNt.max = 4, CNn = 1 )
```

windowValues *Bins sequencing data for plotting*

Description

Given a variable with corresponding genomic positions, the function bins the values in windows of a specified size and calculates weighted mean and 25th and 75th percentile for each window. The resulting object are visualized by the function `plotWindows`.

Usage

```
windowValues(x, positions, chromosomes, window = 1e6, overlap = 0,
    weight = rep.int( x = 1, times = length(x)), start.coord = 1)
windowBf(Af, Bf, good.reads, positions, chromosomes, window = 1e6,
    overlap = 0, start.coord = 1, conf = 0.95)
```

Arguments

`x` variable to be windowed.

`positions` base-pair positions.

`chromosomes` names or numbers of the chromosomes.

`window` size of windows used for binning data. Smaller windows will take more time to compute.

`overlap` integer defining the number of overlapping windows. Default is 0, no overlap.

`weight` weights to be assigned to each value of `x`, usually related to the read depth.

`start.coord` coordinate at which to start computing the windows. If NULL, will start at the first position available.

`Af` A-allele frequency for the `Bf` calculation.

`Bf` B-allele frequency for the `Bf` calculation.

`good.reads` number of reads passing filter for the `Bf` calculation.

`conf` confidence intervals of the binned `Bf` value.

Details

DNA sequencing produces an amount of data too large to be handled by standard graphical devices. In addition, for samples analyzed with older machines and with low or middle coverage (20x to 50x), measures such as read depth are subject to big variations due to technical noise. Using `windowValues` prior to plotting reduces the noise and the amount of data to be plotted.

The binning of the B-allele frequency requires a separate function, `windowBf`, as the B-allele frequency calculation uses multiple values: `Af`, `Bf` and `good.reads`.

The output of `windowValues` and `windowBf` can be used as input for `plotWindows`.

Value

a list of data.frame, one per chromosome. Each data.frame contains base-pair windows covering the chromosome. Each row of the data.frame correspond to a window and its weighted mean, 25th and 75th percentiles of the input values, and the number of data points within each window.

See Also

`plotWindows`

Examples

```
## Not run:
    data.file <-  system.file("extdata", "example.seqz.txt.gz",
        package = "sequenza")
    seqz.data <- read.seqz(data.file)
    # 1Mb windows, each window is overlapping with 1 other
    # adjacent window: depth ratio
    seqz.ratio <- windowValues(x = seqz.data$depth.ratio,
        positions = seqz.data$position, chromosomes = seqz.data$chromosome,
        window = 1e6, weight = seqz.data$depth.normal, start.coord = 1,
        overlap = 1)

    seqz.hom  <- seqz.data$zygosity.normal == 'hom'
    seqz.het  <- seqz.data[!seqz.hom, ]
    # 1Mb windows, each window is overlapping with 1 other adjacent window:
    # B-allele frequency
    seqz.bafs  <- windowValues(x = seqz.het$Bf, positions = seqz.het$position,
        chromosomes = seqz.het$chromosome, window = 1e6,
        weight = seqz.het$depth.tumor, start.coord = 1, overlap = 1)
    # Repeat the same operation using windowBf
    seqz.bafs  <- windowBf(Af = seqz.het$Bf, Bf = seqz.het$Bf,
        good.reads = seqz.het$good.reads, positions = seqz.het$position,
        chromosomes = seqz.het$chromosome, window = 1e6,
        start.coord = 1, overlap = 1, conf = 0.95)

## End(Not run)
```

Index