Package 'copynumber'

March 19, 2018

Title Segmentation of single- and multi-track copy number data by penalized least squares regression.

Version 1.18.0

Author Gro Nilsen, Knut Liestoel and Ole Christian Lingjaerde.

Maintainer Gro Nilsen <gronilse@ifi.uio.no>

Description Penalized least squares regression is applied to fit piecewise constant curves to copy number data to locate genomic regions of constant copy number. Procedures are available for individual segmentation of each sample, joint segmentation of several samples and joint segmentation of the two data tracks from SNP-arrays. Several plotting functions are available for visualization of the data and the segmentation results.

License Artistic-2.0

LazyData yes

Date 2013-04-16

BuildResaveData best

Depends R (>= 2.10), BiocGenerics

Imports S4Vectors, IRanges, GenomicRanges

biocViews aCGH, SNP, CopyNumberVariation, Genetics, Visualization

NeedsCompilation no

R topics documented:

`aspcf` *Allele-specific copy number segmentation.*

Description

Joint segmentation of SNP array data resulting in piecewise constant curves with common break points for copy number data and B-allelle frequency data.

Usage

```
aspcf(logR, BAF, pos.unit = "bp", arms = NULL, kmin = 5, gamma = 40,
      baf.thres=c(0.1,0.9), skew = 3, assembly= "hg19", digits = 4,
      return.est = FALSE, save.res = FALSE, file.names=NULL, verbose = TRUE)
```

Arguments

`logR`	either a data frame or the name of a tab-separated file from which copy number data can be read. The rows of the data frame or file should represent the probes. Column 1 must hold numeric or character chromosome numbers, column 2 the numeric local probe positions, and subsequent columns the numeric copy number measurements for one or more samples. The header of copy number column(s) should give sample ID(s).
`BAF`	either a data frame or the name of a tab-separated file from which B-allelle frequency data can be read. Must be on the same format and size as `logR`, with chromosomes and local probe positions in the two first columns, and numeric BAF-measurements for one or more samples in subsequent columns.
`pos.unit`	the unit used to represent the probe positions. Allowed options are "mbp" (mega base pairs), "kbp" (kilo base pairs) or "bp" (base pairs). By default assumed to be "bp".
`arms`	optional character vector containing chromosome arms (denoted 'p' and 'q') corresponding to the chromosomes and positions found in `logR` and `BAF`. If not specified chromosome arms are found using the built-in genome assembly version determined by `assembly`.
`kmin`	minimum number of probes in each segment, default is 5.
`gamma`	penalty for each discontinuity in the curve, default is 40.

baf.thres	a numeric vector of length two giving the thresholds below and above which BAF probes are considered germline homozygous. Must be in the range 0 to 1, default is 0.1 and 0.9 for the lower and upper limit, respectively.
skew	a numeric value used to determine whether there is allelic skewness (one or two bands) in BAF. Default is 3. The larger the value the further the BAF measurements must be from 0.5 to imply two bands.
assembly	a string specifying which genome assembly version should be applied to determine chromosome arms. Allowed options are "hg19", "hg18", "hg17" and "hg16" (corresponding to the four latest human genome annotations in the UCSC genome browser).
digits	the number of decimals to be applied when reporting results. Default is 4.
return.est	logical value indicating whether a data frame holding LogR estimates should be returned along with the segments. Default is FALSE, which means that only segments are returned.
save.res	logical value indicating whether results should be saved in text files, default is FALSE.
file.names	optional character vector of length two giving the name of the files where the logR estimates and segments, respectively, should be saved in case save.res=TRUE.
verbose	logical value indicating whether or not to print a progress message each time aspcf analysis is finished for a new chromosome arm.

Details

Piecewise constant curves are simultaneously fitted to the LogR and BAF data as described in Nilsen and Liestoel et al.(2012). This implies that break points will be the same for the LogR and BAF segmentation curves, while segment values differ. Segmentation is done separately on each chromosome arm in each sample.

Value

If return.est = TRUE a list with the following components:

logR_estimates	a data frame where the first two columns give the chromosome numbers and probe positions, respectively, while subsequent column(s) give the LogR estimates for each sample. The estimate for a given probe equals the mean of the segment where the probe is located.
segments	a data frame describing each segment found. Each row represents a segment, and columns give the sample IDs, chromosome numbers, arms, local start positions, local end positions, number of probes in the segments, mean LogR values and mean BAF values, respectively.

If return.est = FALSE, only the data frame containing the segments is returned.

If save.res = TRUE the results are also saved in text files with names as specified in file.names. If file.names=NULL, a folder named "aspcf_results" is created in the working directory, and the LogR estimates and the segmentation results are saved in this folder as tab-separated files named logR_estimates.txt and segments.txt, respectively.

Note

It will usually be advisable to Winsorize the logR data before running aspcf, see winsorize on this. Missing values are not allowed in logR, see imputeMissing for imputation of missing copy number values.

Author(s)

Gro Nilsen, Knut Liestoel, Ole Christian Lingjaerde

References

Nilsen and Liestoel et al., "Copynumber: Efficient algorithms for single- and multi-track copy number segmentation", BMC Genomics 13:591 (2012), doi:10.1186/1471-2164-13-59

See Also

plotAllele, winsorize

Examples

```
#Load LogR and BAF data:
data(logR)
data(BAF)

#First winsorize logR to handle outliers:
wins.logR <- winsorize(logR)

#Run aspcf:
aspcf.segments <- aspcf(wins.logR,BAF)
```

---

callAberrations *Call aberrations in segmented data*

Description

Segments, obtained by pcf or multipcf, are classified as "gain", "normal" or "loss" given the specified thresholds.

Usage

```
callAberrations(segments, thres.gain, thres.loss = -thres.gain)
```

Arguments

segments	a data frame containing the segmentation results found by either pcf or multipcf.
thres.gain	a numeric value giving the threshold to be applied for calling gains.
thres.loss	a numeric value giving the threshold to be applied for calling losses. Default is to use the negative value of thres.gain.

Details

Each region found in segments is classified as "gain", "normal" or "loss". Regions with gain or loss will be those segments where the segment value is above or below the value given in thres.gain or thres.loss, respectively.

Value

A new segment data frame where the segment values have been replaced by the classification "gain", "normal" or "loss".

Author(s)

Gro Nilsen

Examples

```
#load lymphoma data
data(lymphoma)
#Run pcf
seg <- pcf(data=lymphoma,gamma=12)

#Call gains as segments whose value is > 0.2, and losses as segments whose
# value < -0.1
ab.seg <- callAberrations(seg,thres.gain=0.2,thres.loss=-0.1)
```

---

getGRangesFormat *Get segments on the GRanges format*

Description

The segments data frame obtained e.g. by `pcf`, `multipcf` or `aspcf` is converted to the GRanges format.

Usage

```
getGRangesFormat(segments)
```

Arguments

`segments`	a data frame containing segmentation results found by e.g. `pcf`, `multipcf` or `aspcf`.

Details

GRanges, in the GenomicRanges package, is the standard BioConductor containers for range data. For some applications it may therefore be useful to convert segmentation results to this format.

Value

The segments converted to the GRanges container class.

Author(s)

Gro Nilsen

Examples

```
#load lymphoma data
data(lymphoma)
#Run pcf
seg <- pcf(data=lymphoma,gamma=12)
#Obtain the GRanges format
gr <- getGRangesFormat(seg)
```

`imputeMissing` *Impute missing copy number values*

Description

Missing copy number values are imputed by a constant value or pcf-estimates.

Usage

```
imputeMissing(data, method, c = 0, pcf.est = NULL,...)
```

Arguments

`data`	a data frame with numeric or character chromosome numbers in the first column, numeric local probe positions in the second, and numeric copy number data for one or more samples in subsequent columns.
`method`	the imputation method to be used. Must be one of "constant" and "pcf".
`c`	a numerical value to be imputed if method is "constant". Default is 0.
`pcf.est`	a data frame of same size as `data`, with chromosome numbers and positions in the first two columns, and copy number estimates obtained from `pcf` in the subsequent columns. Only applicable if `method="pcf"`. If unspecified and `method="pcf"`, `pcf` is run internally to find estimates.
`...`	other relevant parameters to be passed on to `pcf`

Details

The available imputation methods are:

`constant:` all missing values in `data` are replaced by the specified value `c`.

`pcf:` the estimates from pcf-segmentation (see `pcf`) are used to impute missing values. If `pcf` has already been run, these estimates may be specified in `pcf.est`. If `pcf.est` is unspecified, `pcf` is run on the input data. In `pcf` the analysis is done on the observed values, and estimates for missing observations are set to be the estimate of the nearest observed probe.

Value

A data frame of the same size and format as `data` with all missing values imputed.

Author(s)

Gro Nilsen

See Also

pcf

Examples

```
#Load lymphoma data
data(lymphoma)
chrom <- lymphoma[,1]
pos <- lymphoma[,2]
#pick out data for the first six samples:
cn.data <- lymphoma[,3:8]

#Create missing values in cn.data at random positions:
n <- nrow(cn.data)*ncol(cn.data)
r <- matrix(rbinom(n=n,size=1,prob=0.95),nrow=nrow(cn.data),ncol=ncol(cn.data))
cn.data[r==0] <- NA    #matrix with approximately 5% missing values
mis.data <- data.frame(chrom,pos,cn.data)

#Impute missing values by constant, c=0:
imp.data <- imputeMissing(data=mis.data,method="constant")

#Impute missing values by obtained pcf-values:
pcf.est <- pcf(data=mis.data,return.est=TRUE)
imp.data <- imputeMissing(data=mis.data,method="pcf",pcf.est=pcf.est)

#Or run pcf within imputeMissing:
imp.data <- imputeMissing(data=mis.data,method="pcf")
```

---

interpolate.pcf *Interpolation of pcf-estimates.*

Description

Given a segmentation by pcf, interpolate pcf-estimates for specific positions.

Usage

```
interpolate.pcf(segments,x)
```

Arguments

segments	a data frame containing the segmentation results from pcf.
x	matrix or data.frame where the first column gives chrosomomes and the second gives positions.

Details

Pcf-estimates are interpolated for the chromosomes and postions specified in x.

Value

A data frame where the first two columns give the chromsomes and positions specified in the input `x` and the remaining columns give the interpolated pcf-estimate for each sample represented in `segments`.

Note

The positions in `segments` and `x` must be of the same unit (bp, kbp, or mbp).

Author(s)

Gro Nilsen, Ole Christian Lingjaerde.

See Also

`pcf`

Examples

```
#Load the lymphoma data set:
data(lymphoma)

#Take out a smaller subset of 3 samples (using subsetData):
sub.lymphoma <- subsetData(lymphoma,sample=1:3)

#Run pcf:
seg <- pcf(data=sub.lymphoma,gamma=12)

#Make a matrix with two positions and chromosomes for which we want to
#interpolate the pcf-estimate:
pos <-  c(2000000,50000000)
chr <- c(1,2)
x <- cbind(chr,pos)

#Interpolate
int.pcf <- interpolate.pcf(seg,x)
```

---

lymphoma *3K aCGH data*

Description

A subset of the aCGH data set taken from the reference below.

Usage

```
data(lymphoma)
```

Format

Data frame containing 3091 probes with log2-ratio copy numbers for 21 samples. The first column contains the chromosome numbers, the second gives the local probe positions (in base pairs), while the subsequent columns contain the copy number measurements for the individual samples.

Source

Eide et al., "Genomic alterations reveal potential for higher grade transformation in follicular lymphoma and confirm parallel evolution of tumor cell clones", Blood 116:1489-1497, 2010

Examples

```
#Get data
data(lymphoma)
```

---

micma *Subset of 244K aCGH data*

Description

A subset of the 244K MicMa data set containing copy number measurements for six samples on chromosome 17.

Usage

```
data(micma)
```

Format

Data frame containing 7658 probes with log2-ratio copy numbers for 6 samples on chromosome 17. The first column contains the chromosome numbers, the second gives the local probe positions (in base pairs), while the subsequent columns contain the copy number measurements for the individual samples.

Source

Mathiesen et al., "High resolution analysis of copy number changes in disseminated tumor cells of patients with breast cancer", Int J Cancer 131(4):E405:E415, 2011

Examples

```
#Get data
data(micma)
```

---

multipcf *Multi-sample copy number segmentation.*

Description

Joint segmentation resulting in piecewise constant curves with common break points for all samples.

Usage

```
multipcf(data, pos.unit = "bp", arms = NULL, Y = NULL, gamma = 40,
          normalize=TRUE, w=1, fast = TRUE, assembly = "hg19", digits = 4,
          return.est = FALSE, save.res = FALSE, file.names = NULL, verbose
          = TRUE)
```

Arguments

data	either a data frame or the name of a tab-separated file from which copy number data can be read. The rows of the data frame or file should represent the probes. Column 1 must hold numeric or character chromosome numbers, column 2 the numeric local probe positions, and subsequent columns the numeric copy number measurements for two or more samples. The header of copy number columns should give sample IDs.
pos.unit	the unit used to represent the probe positions. Allowed options are "mbp" (mega base pairs), "kbp" (kilo base pairs) or "bp" (base pairs). By default assumed to be "bp".
arms	optional character vector containing chromosome arms (denoted 'p' and 'q') corresponding to the chromosomes and positions found in data. If not specified chromosome arms are found using the built-in genome assembly version determined by assembly.
Y	either a data frame or the name of a tab-separated file containing original copy number data in the case where data contains Winsorized values. If provided, these values are used to calculate the mean of each segment, otherwise the copy number values in data are used. Y must be on the same form as data.
gamma	penalty for each discontinuity in the curve, default is 40.
normalize	a logical value indicating whether each sample's copy number measurements should be scaled by the sample specific residual standard error. Default is TRUE.
w	a numeric vector giving an individual weight to be used for each sample. May be of length 1 if the same weight should be applied for each sample, default is 1 (no weighting).
fast	a logical value indicating whether a fast (not guaranteed to be exact) version should be run on chromosome arms with > 400 probes. Default is TRUE.
assembly	a string specifying which genome assembly version should be applied to determine chromosome arms. Allowed options are "hg19", "hg18", "hg17" and "hg16" (corresponding to the four latest human genome annotations in the UCSC genome browser).
digits	the number of decimals to be applied when reporting results. Default is 4.
return.est	logical value indicating whether a data frame with copy number estimates (multipcf estimates)should be returned along with the segments. Default is FALSE, which means that only segments are returned.
save.res	logical value indicating whether results should be saved in text files, default is FALSE.
file.names	optional character vector of length two giving the name of the files where the multipcf estimates and segments, respectively, should be saved in case save.res = TRUE.
verbose	logical value indicating whether or not to print a progress message each time multipcf analysis is finished for a new chromosome arm.

Details

Piecewise constant curves are simultaneously fitted to the copy number data for several samples as described in the multiPCF algorithm in Nilsen and Liestoel et al. (2012). This implies that break points will be the same for all segmentation curves, but the mean segment values will differ among samples. Segmentation is done separately on each chromosome arm.

Value

If `return.est = TRUE` a list with the following components:

`estimates` a data frame where the first two columns give the chromosome numbers and probe positions, respectively, while subsequent columns give the copy number estimates for each sample. The estimate for a given probe and sample equals the sample mean of the segment where the probe is located.

`segments` a data frame describing the segments found in the data. Each row represents a segment, and the first five columns give the chromosome numbers, arms, local start positions, local end positions, and the number of probes in the segments, respectively. Subsequent columns give the mean segment value for each sample, with sample IDs as column headers.

If `return.est = FALSE` only the data frame containing the segments is returned.

If `save.res = TRUE` the results are also saved in text files with names as specified in `file.names`. If `file.names=NULL`, a folder named "multipcf_results" is created in the working directory, and the segments and copy number estimates are saved in this folder as tab-separated files named segments.txt and estimates.txt, respectively.

Note

It is usually advisable to Winsorize data before running pcf, see `winsorize` on this.

The input data must be complete, see `imputeMissing` for imputation of missing copy number values.

Author(s)

Gro Nilsen, Knut Liestoel

References

Nilsen and Liestoel et al., "Copynumber: Efficient algorithms for single- and multi-track copy number segmentation", BMC Genomics 13:591 (2012), doi:10.1186/1471-2164-13-59

See Also

`imputeMissing`, `pcf`

Examples

```
#Load lymphoma data:
data(lymphoma)

#Take out a subset of 3 biopsies from the first patient (using subsetData):
sub.lymphoma <- subsetData(lymphoma,sample=1:3)

#Check for missing values in data:
any(is.na(sub.lymphoma))
#FALSE

#First winsorize data to handle outliers:
wins.lymph <- winsorize(sub.lymphoma)

#Run multipcf on subset lymphoma data (using a low gamma because of low-density data)
```

```
multi.segments <- multipcf(data=wins.lymph,gamma=12,Y=sub.lymphoma)
```

---

pcf *Single-sample copy number segmentation.*

Description

Fit a individual piecewise constant segmentation curve to each sample's copy number data.

Usage

```
pcf(data, pos.unit = "bp", arms = NULL, Y = NULL, kmin = 5, gamma = 40,
      normalize = TRUE, fast = TRUE, assembly = "hg19", digits = 4,
      return.est = FALSE, save.res = FALSE, file.names = NULL, verbose = TRUE)
```

Arguments

data	either a data frame or the name of a tab-separated file from which copy number data can be read. The rows of the data frame or file should represent the probes. Column 1 must hold numeric or character chromosome numbers, column 2 the numeric local probe positions, and subsequent column(s) the numeric copy number measurements for one or more samples. The header of copy number columns should give sample IDs.
pos.unit	the unit used to represent the probe positions. Allowed options are "mbp" (mega base pairs), "kbp" (kilo base pairs) or "bp" (base pairs). By default assumed to be "bp".
arms	optional character vector containing chromosome arms (denoted 'p' and 'q') corresponding to the chromosomes and positions found in data. If not specified chromosome arms are found using the built-in genome assembly version determined by assembly.
Y	either a data frame or the name of a tab-separated file containing original copy number data in the case where data contains Winsorized values. If provided, these values are used to calculate the mean of each segment, otherwise the copy number values in data are used. Y must be on the same form as data.
kmin	minimum number of probes in each segment, default is 5.
gamma	penalty for each discontinuity in the curve, default is 40.
normalize	logical value indicating whether the copy number measurements should be scaled by the sample residual standard error. Default is TRUE.
fast	a logical value indicating whether a fast (not guaranteed to be exact) version should be run on chromosome arms with > 400 probes.
assembly	a string specifying which genome assembly version should be applied to determine chromosome arms. Allowed options are "hg19", "hg18", "hg17" and "hg16" (corresponding to the four latest human genome annotations in the UCSC genome browser).
digits	the number of decimals to be applied when reporting results. Default is 4.
return.est	logical value indicating whether a data frame holding copy number estimates (pcf values) should be returned along with the segments. Default is FALSE, which means that only segments are returned.

save.res	logical value indicating whether results should be saved in text files.
file.names	optional character vector of length two giving the name of the files where the pcf estimates and segments, respectively, should be saved in case save.res=TRUE.
verbose	logical value indicating whether or not to print a progress message each time pcf analysis is finished for a new chromosome arm.

Details

A piecewise constant segmentation curve is fitted to the copy number observations as described in the PCF algorithm in Nilsen and Liestoel et al. (2012). Segmentation is done separately on each chromosome arm in each sample.

Value

If return.est = TRUE a list with the following components:

estimates	a data frame where the first two columns give the chromosome numbers and probe positions respectively, while subsequent column(s) give the copy number estimates for each sample. The estimate for a given probe equals the mean of the segment where the probe is located.
segments	a data frame describing each segment found in the data. Each row represents a segment, while columns give the sampleID, chromosome number, arm, local start position, local end position, number of probes in the segment and mean value, respectively.

If return.est = FALSE, only the data frame containing the segments is returned.

If save.res = TRUE the results are also saved in text files with names as specified in file.names. If file.names=NULL, a folder named "pcf_results" is created in the working directory, and the pcf estimates and segments are saved in this directory in tab-separated files named estimates.txt and segments.txt, respectively.

Note

It is usually advisable to Winsorize data before running pcf, see winsorize on this.

Missing copy number values are allowed. These are kept out of the pcf analysis, and copy number estimates for missing observations are later set to be the same as the estimate of the nearest observed probe.

Author(s)

Gro Nilsen, Knut Liestoel, Ole Christian Lingjaerde.

References

Nilsen and Liestoel et al., "Copynumber: Efficient algorithms for single- and multi-track copy number segmentation", BMC Genomics 13:591 (2012), doi:10.1186/1471-2164-13-59

See Also

multipcf

Examples

```
#Load the lymphoma data set:
data(lymphoma)

#Take out a smaller subset of 3 samples (using subsetData):
sub.lymphoma <- subsetData(lymphoma,sample=1:3)

#First winsorize data to handle outliers:
wins.lymph <- winsorize(sub.lymphoma)

#Run pcf (using small gamma because of low-density data):
pcf.segments <- pcf(data=wins.lymph,gamma=12,Y=sub.lymphoma)
```

---

pcfPlain *Plain single-sample copy number segmentation.*

Description

A basic single-sample pcf segmentation which does not take chromosome borders into account

Usage

```
pcfPlain(pos.data, kmin = 5, gamma = 40, normalize = TRUE, fast = TRUE,
      digits = 4, return.est = FALSE, verbose = TRUE)
```

Arguments

pos.data	a data frame where the rows represent the probes, column 1 holds probe positions, and subsequent column(s) give the numeric copy number measurements for one or more samples. The header of copy number columns should give sample IDs.
kmin	minimum number of probes in each segment, default is 5.
gamma	penalty for each discontinuity in the curve, default is 40.
normalize	logical value indicating whether the copy number measurements should be scaled by the sample residual standard error. Default is TRUE.
fast	a logical value indicating whether a fast (not guaranteed to be exact) version should be run if the number of probes are > 400.
digits	the number of decimals to be applied when reporting results. Default is 4.
return.est	logical value indicating whether a data frame holding copy number estimates (pcf values) should be returned along with the segments. Default is FALSE, which means that only segments are returned.
verbose	logical value indicating whether or not to print a progress message each time pcf analysis is finished for a sample.

Details

A piecewise constant segmentation curve is fitted to the copy number observations as described in the PCF algorithm in Nilsen and Liestoel et al. (2012). Unlike the regular `pcf` function, `pcfPlain` does not make independent segmentations for each chromosome arm (i.e. breakpoints are not automatically inserted at the beginning and end of chromosome arms). The segmentation can thus be performed independently of assembly.

Value

If `return.est = TRUE` a list with the following components:

`estimates`	a data frame where the first column gives the probe positions, while subsequent column(s) give the copy number estimates for each sample. The estimate for a given probe equals the mean of the segment where the probe is located.
`segments`	a data frame describing each segment found in the data. Each row represents a segment, while columns give the sampleID, start position, end position, number of probes in the segment and mean value, respectively.

If `return.est = FALSE`, only the data frame containing the segments is returned.

Note

If probe positions are not available, the first column in `data` may, e.g., contain the values `1:nrow(data)`.

Author(s)

Gro Nilsen, Knut Liestoel, Ole Christian Lingjaerde.

References

Nilsen and Liestoel et al., "Copynumber: Efficient algorithms for single- and multi-track copy number segmentation", BMC Genomics 13:591 (2012), doi:10.1186/1471-2164-13-59

See Also

`pcf`

Examples

```
#Load the lymphoma data set:
data(lymphoma)

#Take out a smaller subset of 3 samples (using subsetData):
sub.lymphoma <- subsetData(lymphoma,sample=1:3)

#Run pcfPlain (remove first column of chromosome numbers):
plain.segments <- pcfPlain(pos.data=sub.lymphoma[,-1],gamma=12)
```

plotAberration *Plot areas with copy number aberrations*

Description

Create plots reflecting the location of aberrated segments. Results may be visualized over the entire genome or by chromosomes.

Usage

```
plotAberration(segments, thres.gain, thres.loss = -thres.gain, pos.unit = "bp",
    chrom = NULL, layout = c(1, 1),...)
```

Arguments

segments	a data frame containing the segmentation results found by either pcf or multipcf.
thres.gain	a numeric vector giving the threshold value(s) to be applied for calling gains.
thres.loss	a numeric vector of same length as thres.gain giving the threshold value(s) to be applied for calling losses. Default is to use the negative value of thres.gain.
pos.unit	the unit used to represent the probe positions. Allowed options are "mbp" (mega base pairs), "kbp" (kilo base pairs) or "bp" (base pairs). By default assumed to be "bp".
chrom	a numeric or character vector with chromosome number(s) to indicate which chromosome(s) is (are) to be plotted. If unspecified the whole genome is plotted.
layout	the vector of length two giving the number of rows and columns in the plot window. Default is c(1,1).
...	other optional graphical parameters. These include the plot arguments xlab, ylab, main, cex.main, mgp, cex.lab, cex.axis, mar and title (see par on these), as well as plot.size, plot.unit, plot.ideo, ideo.frac, cyto.text, assembly and cex.cytotext (see plotSample on these). In addition, a range of graphical arguments specific for this plot function may be specified:
	colors a character vector of length two where the first and second element specifies the color used to represent loss and gain, respectively. Default is c("dodgerblue","red").
	sample.labels a logical value indicating whether sample labels are to be plotted along the y-axis. Default is TRUE.
	sep.samples a number in the range 0 to 0.4 used to create some space between samples. 0 implies that there is no space. Default is 2/nsample, where nsample is the number of samples found in segments.
	sample.line a numeric scalar giving the margin line where the sample labels should be written, starting at 0 counting outwards. Default is 0.2.
	sample.cex the size of the sample labels.

Details

For each sample, the aberrated regions are shown in the color specified in colors[1] (default dodgerblue) if the segment value is below thres.loss and the color specified in colors[2] (default red) if the segment value is above thres.gain. Non-aberrated regions are shown in white. Each row in the plot represents a sample, while probe positions are reflected along the x-axis.

Note

This function applies `par(fig)`, and is therefore not compatible with other setups for arranging multiple plots in one device such as `par(mfrow,mfcol)`.

Author(s)

Gro Nilsen

Examples

```
#Load lymphoma data
data(lymphoma)

#Run pcf to obtain estimated copy number values
seg <- pcf(data=lymphoma,gamma=12)

#Plot aberrations for the entire genome
plotAberration(segments=seg,thres.gain=0.15)

#Plot aberrations for the first 4 chromosomes:
plotAberration(segments=seg,thres.gain=0.1,chrom=c(1:4),layout=c(2,2))
```

---

plotAllele *Plot SNP data and/or aspcf segmentation results*

Description

Plot bivariate SNP data and/or aspcf segmentation results for each sample separately with chromosomes in different panels

Usage

```
plotAllele(logR = NULL, BAF = NULL, segments = NULL, pos.unit = "bp",
            sample = NULL, chrom = NULL, assembly="hg19", baf.thres =
            c(0.1,0.9), winsoutliers = NULL, xaxis = "pos", layout = c(1,1),
            plot.ideo = TRUE, ...)
```

Arguments

`logR`	a data frame with numeric or character chromosome numbers in the first column, numeric local probe positions in the second, and numeric copy number data for one or more samples in subsequent columns. The header of the copy number column(s) should give the sample IDss.
`BAF`	a data frame on the same format and size as `logR`, with chromosomes and local probe positions in the two first columns, and numeric BAF-measurements for one or more samples in subsequent columns.
`segments`	a data frame or a list of data frames containing the segmentation results found by `aspcf`.

pos.unit	the unit used to represent the probe positions. Allowed options are "mbp" (mega base pairs), "kbp" (kilo base pairs) or "bp" (base pairs). By default assumed to be "bp".
sample	a numeric vector indicating which sample(s) is (are) to be plotted. The number(s) should correspond to the sample's place (in order of appearance) in logR, or in segments if logR is unspecified.
chrom	a numeric or character vector with chromosome number(s) to indicate which chromosome(s) is (are) to be plotted.
assembly	a string specifying which genome assembly version should be applied to define the chromosome ideogram. Allowed options are "hg19", "hg18", "hg17" and "hg16" (corresponding to the four latest human genome annotations in the UCSC genome browser).
baf.thres	a numeric vector of length 2 giving thresholds below/above which BAF-values will not be plotted (use this to remove germline homozygous BAF probes from the plot).
winsoutliers	an optional data frame of the same size as logR identifying observations classified as outliers by winsorize. If specified, outliers will be marked by a different color and symbol than the other observations (see wins.col and wins.pch).
xaxis	either "pos" or "index". The former implies that the xaxis will represent the genomic positions, whereas the latter implies that the xaxis will represent the probe indices. Default is "pos".
layout	an integer vector of length two giving the number of rows and columns in the plot. Default is c(1,1).
plot.ideo	a logical value indicating whether the chromosome ideogram should be plotted. Only applicable when xaxis="pos".
...	other graphical parameters. These include the common plot arguments xlab, ylab, main, xlim, ylim, col (default is "grey"), pch (default is 46, equivalent to "."), cex, cex.lab, cex.main, cex.axis, las, tcl, mar and mgp (see par on these). In addition, a range of graphical arguments specific for copy number plots may be specified, see plotSample on these.

Details

Several chromosome may be displayed on the same page with the layout option. If the number of chromosomes exceeds the desired page layout, the user is prompted before advancing to the next page of output.

Note

This function applies par(fig), and is therefore not compatible with other setups for arranging multiple plots in one device such as par(mfrow,mfcol).

Author(s)

Gro Nilsen

Examples

```
#Load logR and BAF data:
data(logR)
data(BAF)
```

```
#Run aspcf::
aspcf.segments <- aspcf(logR,BAF)

#Plot
plotAllele(logR,BAF,aspcf.segments,layout=c(2,2))
```

---

plotChrom *Plot copy number data and/or segmentation results by chromosome*

Description

Plot copy number data and/or segmentation results for each chromosome separately with samples in different panels.

Usage

```
plotChrom(data = NULL, segments = NULL, pos.unit = "bp", sample = NULL,
            chrom = NULL, assembly = "hg19", winsoutliers = NULL, xaxis =
            "pos", layout = c(1,1), plot.ideo = TRUE, ...)
```

Arguments

data	a data frame with numeric or character chromosome numbers in the first column, numeric local probe positions in the second, and numeric copy number data for one or more samples in subsequent columns. The header of the copy number columns should be the sample IDs.
segments	a data frame or a list of data frames containing the segmentation results found by either pcf or multipcf.
pos.unit	the unit used to represent the probe positions. Allowed options are "mbp" (mega base pairs), "kbp" (kilo base pairs) or "bp" (base pairs). By default assumed to be "bp".
sample	a numeric vector indicating which sample(s) is (are) to be plotted. The number(s) should correspond to the sample's place (in order of appearance) in data, or in segments in case data is unspecified.
chrom	a numeric or character vector with chromosome number(s) to indicate which chromosome(s) is (are) to be plotted.
assembly	a string specifying which genome assembly version should be applied to define the chromosome ideogram. Allowed options are "hg19", "hg18", "hg17" and "hg16" (corresponding to the four latest human genome annotations in the UCSC genome browser).
winsoutliers	an optional data frame of the same size as data identifying observations classified as outliers by winsorize. If specified, outliers will be marked by a different color and symbol than the other observations (see wins.col and wins.pch).
xaxis	either "pos" or "index". The former implies that the xaxis will represent the genomic positions, whereas the latter implies that the xaxis will represent the probe index. Default is "pos".
layout	an integer vector of length two giving the number of rows and columns in the plot. Default is c(1,1).

plot.ideo	a logical value indicating whether the chromosome ideogram should be plotted. Only applicable when xaxis="pos".
...	other graphical parameters. These include the common plot arguments xlab, ylab, main, xlim, ylim, col (default is "grey"), pch (default is 46, equivalent to "."), cex, cex.lab, cex.main, cex.axis, las, tcl, mar and mgp (see par on these). In addition, a range of graphical arguments specific for copy number plots may be specified, see plotSample on these.

Details

Several plots may be produced on the same page with the layout option. If the number of plots exceeds the desired page layout, the user is prompted before advancing to the next page of output.

Note

This function applies par(fig), and is therefore not compatible with other setups for arranging multiple plots in one device such as par(mfrow,mfcol).

Author(s)

Gro Nilsen

See Also

plotSample, plotGenome

Examples

```
#Lymphoma data
data(lymphoma)
#Take out a smaller subset of 6 samples (using subsetData):
sub.lymphoma <- subsetData(lymphoma,sample=1:6)

#Winsorize data:
wins.res <- winsorize(data=sub.lymphoma,return.outliers=TRUE)

#Use pcf to find segments:
uni.segments <- pcf(data=wins.res,gamma=12)

#Use multipcf to find segments as well:
multi.segments <- multipcf(data=wins.res,gamma=12)

#Plot data and segments for chromosome 1 separately for each sample:
plotChrom(data=sub.lymphoma,segments=list(uni.segments,multi.segments),chrom=1,
    layout=c(3,2))
#Let xaxis be probe index, and do not connect segments by vertical lines:
plotChrom(data=sub.lymphoma,segments=list(uni.segments,multi.segments),chrom=1,
    xaxis="index",layout=c(3,2),legend=FALSE,connect=FALSE)
#Data was winsorized earlier. Mark winsorized values by different color
#and symbol:
plotChrom(data=wins.res,chrom=1,winsoutliers=wins.res,layout=c(3,2))
#Save plots in working directory:
plotChrom(data=sub.lymphoma,segments=uni.segments,chrom=c(1,2),
    layout=c(3,2),dir.print=getwd(),file.name=c("chromosome1","chromosome2"),
    onefile=FALSE)
```

`plotCircle` *Plot a circular genome with aberration frequencies and connections between genomic loci added.*

Description

A circular genome is plotted and the percentage of samples that have a gain or a loss at a genomic position is added in the middle of the circle. Gains/losses correspond to copy number values that are above/below a pre-defined threshold. In addition arcs representing some connection between genomic loci may be added.

Usage

```
plotCircle(segments, thres.gain, thres.loss = -thres.gain, pos.unit = "bp",
    freq.colors = c("red","limegreen"), alpha = 1/7, arcs = NULL,
    arc.colors = c("goldenrod1","dodgerblue"), d = 0.3, assembly = "hg19")
```

Arguments

`segments`	a data frame containing the segmentation results found by either `pcf` or `multipcf`.
`thres.gain`	a scalar giving the threshold value to be applied for calling gains.
`thres.loss`	a scalar giving the threshold value to be applied for calling losses. Default is to use the negative value of `thres.gain`.
`pos.unit`	the unit used to represent the probe positions. Allowed options are "mbp" (mega base pairs), "kbp" (kilo base pairs) or "bp" (base pairs). By default assumed to be "bp".
`freq.colors`	a vector giving the colors to be used for the amplification and deletion frequencies, respectively. Default is c("red","limegreen").
`alpha`	a scalar in the range 0 to 1 determining the amount of scaling of the aberration frequencies. For the default value of 1/7 the distance between the genome circle and the zero-line of the frequency-circle corresponds to an aberration percentage of 100 %. See details.
`arcs`	an optional matrix or data frame with 5 columns specifying connections between genomic loci. The first two columns must give the chromosome numbers and local positions for the start points of the arcs, while the two next columns give the chromosome numbers and local positions for the end point of arcs. The last column should contain a vector of numbers 1,2,... indicating that the arcs belong to different classes. Each class of arcs will then be plotted in a different color.
`arc.colors`	a vector giving the colors to be used for plotting the different classes of arcs. Cannot be shorter than the number of unique classes in `arcs`. The first color will represent the first class in `arcs`, the second color the second class and so on.
`d`	a scalar > 0 representing the distance from the genome circle to the starting points of the arcs. Set d=0 to make arcs start at the genome circle.
`assembly`	a string specifying which genome assembly version should be applied to determine chromosome ideograms. Allowed options are "hg19", "hg18", "hg17" and "hg16" (corresponding to the four latest human genome annotations in the UCSC genome browser).

Details

To zoom in on the observed aberration frequencies one may increase alpha. However, the user should be aware that this implies that the distance between the genome circle and the frequency zero-line does not reflect an aberration frequency of 100 %. Since the distance between the two circles is always 1/7, the maximum plotted percentage will be 100/(alpha*7) and any percentages that are higher than this will be truncated to this value.

Author(s)

Gro Nilsen

Examples

```
#load lymphoma data
data(lymphoma)
#Run pcf
pcf.res <- pcf(data=lymphoma,gamma=12)

plotCircle(segments=pcf.res,thres.gain=0.1)

#Use alpha to view the frequencies in more detail:
plotCircle(segments=pcf.res,thres.gain=0.1,alpha=1/5)

#An example of how to specify arcs
#Using multipcf, we compute the correlation between all segments and then
#retrieve those that have absolute inter-chromosomal correlation > 0.7
multiseg <- multipcf(lymphoma)
nseg = nrow(multiseg)
cormat = cor(t(multiseg[,-c(1:5)]))
chr.from <- c()
pos.from <- c()
chr.to <- c()
pos.to <- c()
cl <- c()

thresh = 0.7
for (i in 1:(nseg-1)) {
  for (j in (i+1):nseg) {
    #Check if segment-correlation is larger than threshold and that the two
    #segments are located on different chromosomes
    if (abs(cormat[i,j]) > thresh && multiseg$chrom[i] != multiseg$chrom[j]) {
      chr.from = c(chr.from,multiseg$chrom[i])
      chr.to = c(chr.to,multiseg$chrom[j])
      pos.from = c(pos.from,(multiseg$start.pos[i] + multiseg$end.pos[i])/2)
      pos.to = c(pos.to,(multiseg$start.pos[j] + multiseg$end.pos[j])/2)
      if(cormat[i,j] > thresh){
        cl <- c(cl,1)           #class 1 for those with positive correlation
      }else{
        cl <- c(cl,2)           #class 2 for those with negative correlation
      }
    }
  }
}

arcs <- cbind(chr.from,pos.from,chr.to,pos.to,cl)
```

```
#Plot arcs between segment with high correlations; positive correlation in
#orange, negative correlation in blue:
plotCircle(segments=pcf.res,thres.gain=0.15,arcs=arcs,d=0)
```

---

plotFreq *Plot percentage of samples with an aberration at a genomic position*

Description

Plot the percentage of samples that have an amplification or deletion at a genomic position. Amplifications/deletions correspond to copy number values that are above/below a pre-defined threshold. Frequencies may be plotted over the entire genome or separately for each chromosome.

Usage

```
plotFreq(segments, thres.gain, thres.loss = -thres.gain, pos.unit = "bp",
    chrom = NULL, layout = c(1, 1),...)
```

Arguments

segments	a data frame containing the segmentation results found by either pcf or multipcf.
thres.gain	a numeric vector giving the threshold value(s) to be applied for calling gains.
thres.loss	a numeric vector of same length as thres.gain giving the threshold value(s) to be applied for calling losses. Default is to use the negative value of thres.gain.
pos.unit	the unit used to represent the probe positions. Allowed options are "mbp" (mega base pairs), "kbp" (kilo base pairs) or "bp" (base pairs). By default assumed to be "bp".
chrom	a numeric or character vector with chromosome number(s) to indicate which chromosome(s) is (are) to be plotted. If unspecified the whole genome is plotted, otherwise each specified chromosome is plotted in a separate panel.
layout	an integer vector of length two giving the number of rows and columns in the plot. Default is c(1,1).
...	other graphical parameters. These include the common plot arguments xlab, ylab, title, main, cex.main, mgp, cex.lab, cex.axis, ylim, xlim, and las (see par on these), as well as plot.size, plot.unit, plot.ideo, ideo.frac, cyto.text, assembly and cex.cytotext (see plotSample on these). In addition, some other graphical arguments specific for this plot function may be specified: col.gain the color to be used for the gain frequencies, default is "red". col.loss the color to be used for the loss frequencies, default is "blue". continuous a logical value indicating whether the probe frequencies should be presented as continuous, i.e. the plotted probe frequency will start and end halfway between adjacent probes (except across arms when chromosomes are plotted and except across chromosomes when genome is plotted). Default is TRUE. percentLines either a logical value indicating if horizontal percentages lines should be plotted, or a numeric vector with percentages at which such lines should be plotted. Default is TRUE.

Details

The percentage of samples with an aberration is calculated and plotted for all genomic positions. Regions with gain or loss will be those where copy number values are above or below the values given in `thres.gain` and `thres.loss`, respectively.

Note

This function applies `par(fig)`, and is therefore not compatible with other setups for arranging multiple plots in one device such as `par(mfrow,mfcol)`.

Author(s)

Gro Nilsen

Examples

```
#load lymphoma data
data(lymphoma)
#Run pcf
seg <- pcf(data=lymphoma,gamma=12)

#Plot over entire genome, gain and loss thresholds are 0.1 and -0.1:
plotFreq(segments=seg,thres.gain=0.1)

#Plot by chromosomes, two sets of thresholds:
plotFreq(segments=seg,thres.gain=c(0.1,0.2), thres.loss=c(-0.05,-0.1), chrom=c(1:23),
layout=c(5,5))
```

---

plotGamma *Plot segmentation results for several values of gamma*

Description

Data for one sample on one chromosome is segmented by `pcf` for 10 values of gamma, and results are visualized in a multi-grid plot.

Usage

```
plotGamma(data, pos.unit = "bp", gammaRange = c(10,100), dowins = TRUE,
          sample = 1, chrom = 1, cv = FALSE, K = 5, cex = 2, col = "grey",
          seg.col="red", ...)
```

Arguments

`data`	either a data frame or the name of a tab-separated file from which copy number data can be read. The rows of the data frame or file should represent the probes. Column 1 must hold numeric or character chromosome numbers, column 2 the numeric local probe positions, and subsequent column(s) the numeric copy number measurements for one or more samples. The header of copy number columns should give sample IDs.

pos.unit	the unit used to represent the probe positions. Allowed options are "mbp" (mega base pairs), "kbp" (kilo base pairs) or "bp" (base pairs). By default assumed to be "bp".
gammaRange	a vector of length two giving the lowest and highest value of gamma to be applied. 10 (approximately) equally spaced values within this range are applied in the pcf-segmentation. Default range is c(10,100).
dowins	logical value indicating whether data should be winsorized before running pcf. Default is TRUE.
sample	an integer indicating which sample is to be segmented. The number should correspond to the sample's place (in order of appearance) in data. Default is to use the first sample present in the data input.
chrom	a number or character indicating which chromosome is to be segmented. Default is chromosome 1.
cv	logical value indicating whether K-fold cross-validation should be done, see details.
K	the number of folds to use in K-fold cross-validation, default is 5.
cex	size of data points, default is 2.
col	color used to plot data points, default is "grey".
seg.col	color used to plot segments, default is "red".
...	other optional parameters to be passed to pcf.

Details

Data for one sample and one chromosome is selected, and pcf is run on this data subset while applying 10 different gamma-values (within the given range). The output is a multi-grid plot with the data shown in the first panel, the segmentation results for the various gammas in the subsequent 10 panels, and the number of segments found for each gamma in the last panel.

If cv = TRUE a K-fold cross-validation is also performed. For each fold, a random (100/K) per cent of the data are set to be missing, and pcf is run using the different values of gamma. The missing probe values are then predicted by the estimated value of their closest non-missing neighbour (see pcf on this), and the prediction error for this fold is then calculated as the sum of the squared difference between the predicted and the observed values. The process is repeated over the K folds, and the average prediction errors are finally plotted along with the number of segments in the last panel of the plot. The value of gamma for which the minimum prediction error is found is marked by an asterix. Note that such cross-validation tends to favor small values of gamma, and the suitability of the so-called optimal gamma from this procedure should be critically assessed.

Value

If cv = TRUE a list containing:

gamma	the gamma values applied.
pred.error	the average prediction error for each value of gamma.
opt.gamma	the gamma for which the average prediction error is minimized.

Note

This function applies par(fig), and is therefore not compatible with other setups for arranging multiple plots in one device such as par(mfrow,mfcol).

Author(s)

Gro Nilsen, Knut Liestoel, Ole Christian Lingjaerde

See Also

pcf,winsorize

Examples

```
#Micma data
data(micma)

plotGamma(micma,chrom=17)
```

---

plotGenome *Plot copy number data and/or segmentation results*

Description

Plot copy number data and/or segmentation results for the whole genome.

Usage

```
plotGenome(data = NULL, segments = NULL, pos.unit = "bp", sample = NULL,
            assembly="hg19", winsoutliers = NULL, xaxis = "pos",
            layout = c(1,1), ...)
```

Arguments

data	a data frame with numeric or character chromosome numbers in the first column, numeric local probe positions in the second, and numeric copy number data for one or more samples in subsequent columns. The header of the copy number columns should be the sample IDs.
segments	a data frame or a list of data frames containing the segmentation results found by either pcf or multipcf.
pos.unit	the unit used to represent the probe positions. Allowed options are "mbp" (mega base pairs), "kbp" (kilo base pairs) or "bp" (base pairs). By default assumed to be "bp".
sample	a numeric vector indicating which sample(s) is (are) to be plotted. The number(s) should correspond to the sample's place (in order of appearance) in data, or in segments in case data is unspecified.
assembly	a string specifying which genome assembly version should be applied to define the chromosome ideogram. Allowed options are "hg19", "hg18", "hg17" and "hg16" (corresponding to the four latest human genome annotations in the UCSC genome browser).
winsoutliers	an optional data frame of the same size as data identifying observations classified as outliers by winsorize. If specified, outliers will be marked by a different color and symbol than the other observations (see wins.col and wins.pch).

xaxis	either "pos" or "index". The former implies that the xaxis will represent the genomic positions, whereas the latter implies that the xaxis will represent the probe index. Default is "pos".
layout	an integer vector of length two giving the number of rows and columns in the plot. Default is c(1,1).
...	other graphical parameters. These include the common plot arguments xlab, ylab, main, xlim, ylim, col (default is "grey"), pch (default is 46, equivalent to "."), cex, cex.lab, cex.main, cex.axis, las, tcl, mar and mgp (see par on these). In addition, a range of graphical arguments specific for copy number plots may be specified, see plotSample on these.

Details

Several plots may be produced on the same page with the layout option. If the number of plots exceeds the desired page layout, the user is prompted before advancing to the next page of output.

Note

This function applies par(fig), and is therefore not compatible with other setups for arranging multiple plots in one device such as par(mfrow,mfcol).

Author(s)

Gro Nilsen

See Also

plotSample, plotChrom

Examples

```
#Lymphoma data
data(lymphoma)
#Take out a smaller subset of 6 samples (using subsetData):
sub.lymphoma <- subsetData(lymphoma,sample=1:6)

#Winsorize data:
wins.data <- winsorize(data=sub.lymphoma,return.outliers=TRUE)

#Use pcf to find segments:
uni.segments <- pcf(data=wins.data,gamma=12)

#Use multipcf to find segments as well:
multi.segments <- multipcf(data=wins.data,gamma=12)

#Plot data and pcf-segments over entire genome for all six samples (one page
#for each sample):
plotGenome(data=sub.lymphoma,segments=uni.segments)

#Let each sample define its own range, and adjust range to fit all observations:
plotGenome(data=sub.lymphoma,segments=uni.segments,equalRange=FALSE,q=0)

#Add results from multipcf on top for four of the samples and let all plots
#show on one page:
plotGenome(data=sub.lymphoma,segments=list(uni.segments,multi.segments),
```

```
    layout=c(2,2),sample=c(1:4))

#Change segment-colors, line widths, and legend:
plotGenome(data=sub.lymphoma,segments=list(uni.segments,multi.segments),layout=c(2,2),
    seg.col=c("red","blue"),seg.lwd=c(3,2),legend=c("uni","multi")
    ,sample=c(1:4))

#Aberration calling may be done by defining thresholds that determines the cuf-off
#for what should be considered biologically significant aberrations. In this
#example segments which are above 0.2 or below -0.2 are considered aberrated
#regions:
plotGenome(segments=uni.segments,sample=5,connect=FALSE)
abline(h=0.2,col="blue",lty=5)
abline(h=-0.2,col="blue",lty=5)
```

`plotHeatmap` *Plot copy number heatmap*

Description

Heatmap reflecting the magnitude of estimated copy numbers relative to some pre-defined limits. Estimates may be obtained using `pcf` or `multipcf`, and results may be visualized over the entire genome or by chromosomes.

Usage

```
plotHeatmap(segments, upper.lim, lower.lim = -upper.lim, pos.unit = "bp",
    chrom = NULL, layout = c(1, 1),...)
```

Arguments

`segments`	a data frame containing the segmentation results found by either `pcf` or `multipcf`.
`upper.lim`	a positive numeric vector giving the upper limits(s) to be applied. The colors in the heatmap will reflect the magnitude of the estimated copy numbers relative to this limit, see details.
`lower.lim`	a negative numeric vector of same length as `upper.lim` giving the lower limits(s) to be applied. Default is to use the negative value of `upper.lim`.
`pos.unit`	the unit used to represent the probe positions. Allowed options are "mbp" (mega base pairs), "kbp" (kilo base pairs) or "bp" (base pairs). By default assumed to be "bp".
`chrom`	a numeric or character vector with chromosome number(s) to indicate which chromosome(s) is (are) to be plotted. If unspecified the whole genome is plotted.
`layout`	the vector of length two giving the number of rows and columns in the plot window. Default is `c(1,1)`.
`...`	other optional graphical parameters. These include the plot arguments `xlab`, `ylab`, `main`, `cex.main`, `mgp`, `cex.lab`, `cex.axis`, `mar` and `title` (see `par` on these), as well as `plot.size`, `plot.unit`, `plot.ideo`, `ideo.frac`, `cyto.text`, `assembly` and `cex.cytotext` (see `plotSample` on these). In addition, a range of graphical arguments specific for this plot function may be specified:

`colors` a character vector of length three giving the colors to interpolate in the heatmap, default is c("dodgerblue","black","red").

`n.col` an integer giving the number of color shades to be applied in the interpolation, default is 50.

`sample.labels` a logical value indicating whether sample labels are to be plotted along the y-axis. Default is TRUE.

`sep.samples` a number in the range 0 to 0.4 used to create some space between samples. Default is 0, which implies that there is no space.

`sample.line` a numeric scalar giving the margin line where the sample labels should be written, starting at 0 counting outwards. Default is 0.2.

`sample.cex` the size of the sample labels.

Details

For each sample, the segments are represented by a rectangle plotted in a color corresponding to the difference between the segment copy number value and the limits. If the value is below `lower.lim`, the color of the rectangle will equal the input in `colors[1]` (default dodgerblue). If the value is above `lower.lim`, but below zero, the color of the rectangle will be a nuance between the input in `colors[1]` and `colors[2]` (default black). The closer the value is to zero, the closer the nuance will be to `colors[2]`. Similary, if the value is above `upper.lim`, the color of the rectangle will equal the input in `colors[3]` (default red), whereas if the value is below `upper.lim`, but above zero, the color will be a nuance between the input in `colors[2]` and `colors[3]`. Again, the closer the value is to zero, the closer the nuance will be to `colors[2]`.

Each row in the heatmap represents a sample, while probe positions are reflected along the x-axis.

Note

This function applies `par(fig)`, and is therefore not compatible with other setups for arranging multiple plots in one device such as `par(mfrow,mfcol)`.

Author(s)

Gro Nilsen

Examples

```
#Load lymphoma data
data(lymphoma)

#Run pcf to obtain estimated copy number values
seg <- pcf(data=lymphoma,gamma=12)

#Heatmap for entire genome, two limit values:
plotHeatmap(segments=seg,upper.lim=c(0.1,0.5),layout=c(2,1))

#Heatmap for the first 4 chromosomes:
plotHeatmap(segments=seg,upper.lim=0.1,chrom=c(1:4),layout=c(2,2))
```

plotSample *Plot copy number data and/or segmentation results by sample*

Description

Plot copy number data and/or segmentation results for each sample separately with chromosomes in different panels.

Usage

```
plotSample(data = NULL, segments = NULL, pos.unit = "bp", sample = NULL,
           chrom = NULL, assembly = "hg19", winsoutliers = NULL, xaxis =
           "pos", layout = c(1,1), plot.ideo = TRUE, ...)
```

Arguments

data	a data frame with numeric or character chromosome numbers in the first column, numeric local probe positions in the second, and numeric copy number data for one or more samples in subsequent columns. The header of the copy number columns should be the sample IDs.
segments	a data frame or a list of data frames containing the segmentation results found by either pcf or multipcf.
pos.unit	the unit used to represent the probe positions. Allowed options are "mbp" (mega base pairs), "kbp" (kilo base pairs) or "bp" (base pairs). By default assumed to be "bp".
sample	a numeric vector indicating which sample(s) is (are) to be plotted. The number(s) should correspond to the sample's place (in order of appearance) in data, or in segments in case data is unspecified.
chrom	a numeric or character vector with chromosome number(s) to indicate which chromosome(s) is (are) to be plotted.
assembly	a string specifying which genome assembly version should be applied to define the chromosome ideogram. Allowed options are "hg19", "hg18", "hg17" and "hg16" (corresponding to the four latest human genome annotations in the UCSC genome browser).
winsoutliers	an optional data frame of the same size as data identifying observations classified as outliers by winsorize. If specified, outliers will be marked by a different color and symbol than the other observations (see wins.col and wins.pch).
xaxis	either "pos" or "index". The former implies that the xaxis will represent the genomic positions, whereas the latter implies that the xaxis will represent the probe index. Default is "pos".
layout	an integer vector of length two giving the number of rows and columns in the plot. Default is c(1,1).
plot.ideo	a logical value indicating whether the chromosome ideogram should be plotted. Only applicable when xaxis="pos".
...	other graphical parameters. These include the common plot arguments xlab, ylab, main, xlim, ylim, col (default is "grey"), pch (default is 46, equivalent to "."), cex, cex.lab, cex.main, cex.axis, las, tcl, mar and mgp (see par on these). In addition, a range of graphical arguments specific for plotSample (as

well as the similar functions plotChrom, plotGenome and plotAllele) may be specified:

dir.print: an optional directory where the plot(s) is (are) to be saved as pdf file(s). Defaults to NULL which implies that the plot(s) is (are) printed to screen instead.

file.name an optional character vector containing file name(s) for the pdf file(s) to be saved.

onefile: logical value indicating whether all plots should be plotted in one device / saved in one file. Default is TRUE. If FALSE, a new window is opened or a new file is saved for each sample (each chromosome for plotChrom).

plot.size: a numeric vector of length 2 giving the width and height of the plotting window. Default is c(11.6,8.2).

title: an overall title for all plots on one page.

plot.unit: the desired unit to be applied for probe position tick marks along the x-axis. Only "mbp" (default) and "kbp" is allowed.

equalRange: logical value indicating whether the range of the y-axis should be the same across all plots. Defaults to TRUE.

q: a numerical value in the range 0 to 1 indicating that ylim will be set to only include observations between the (1-q/2)- and the (q/2)-quantile. Observations that fall outside these quantiles are truncated to the limits of the plot, and are by default marked by a special symbol (see q.pch). Default is q=0.01 when data is specified, and q=0 otherwise.

q.col, wins.col: colors used to plot truncated observations and outliers. Default is "grey" and "magenta", respectively.

q.pch, wins.pch: symbols used to plot truncated observations and outliers. Default is 42 (equivalent to "*") for both. Note that input must be of the same class as pch (numeric or character).

q.cex, wins.cex: magnification used for truncated observations and outliers relative to cex. Default is 0.4 for both.

h: a numerical value indicating that a horizontal reference line should be plotted at y=h. Default is h=0. h=NULL suppresses the plotting of a reference line.

at.x: the points at which tick-marks on x-axis are to be drawn.

at.y: the points at which tick-marks on y-axis are to be drawn.

main.line: the margin line for the main title.

legend: either a logical value indicating whether legends should be added to the plot if there is more than one segmentation result present in segments, or a character vector giving the legend texts to be used for the segmentation results. Default is TRUE, in which case the legend will be plotted in the topright corner of each plot.

seg.col: color(s) used to plot the segmentation result(s). The default colors are found using the function rainbow(n), where n is the number of segmentation results found in segments (see rainbow for details).

seg.lty: the line type(s) used to plot the segmentation result(s). Default is 1.

seg.lwd: the line width(s) used to plot the segmentation result(s).

connect: logical value indicating whether segments should be connected by vertical lines, default is TRUE.

ideo.frac: a numerical value in the range 0 to 1 indicating the fraction of the plot to be occupied by the chromosome ideogram.

`cyto.text`: a logical value indicating whether cytoband-names should be plotted along with the ideogram. Not recommended when many plots are plotted in the same grid, default is FALSE.

`cex.cytotext`: the magnification used for the plotting of the cytoband-names.

`cex.chrom`: the text size used to plot chromosome numbers in `plotGenome`.

Details

Several plots may be produced on the same page with the `layout` option. If the number of plots exceeds the desired page layout, the user is prompted before advancing to the next page of output.

Note

These functions apply `par(fig)`, and are therefore not compatible with other setups for arranging multiple plots in one device such as `par(mfrow,mfcol)`.

Author(s)

Gro Nilsen

See Also

`plotChrom`, `plotGenome`

Examples

```
#Lymphoma data
data(lymphoma)
#Take out a smaller subset of 6 samples (using subsetData):
sub.lymphoma <- subsetData(lymphoma,sample=1:6)

#Winsorize data:
wins.data <- winsorize(data=sub.lymphoma)

#Use pcf to find segments:
uni.segments <- pcf(data=wins.data,gamma=12)

#Use multipcf to find segments as well:
multi.segments <- multipcf(data=wins.data,gamma=12)

#Plot data and pcf-segments for one sample separately for each chromosome:
plotSample(data=sub.lymphoma,segments=uni.segments,sample=1,layout=c(5,5))
#Add cytoband text to ideogram (one page per chromosome to ensure sufficient
#space)
plotSample(data=sub.lymphoma,segments=uni.segments,sample=1,layout=c(1,1),
    cyto.text=TRUE)
#Add multipcf-segmentation results, drop legend
plotSample(data=sub.lymphoma,segments=list(uni.segments,multi.segments),sample=1,
    layout=c(5,5),seg.col=c("red","blue"),seg.lwd=c(3,2),legend=FALSE)
#Plot by chromosome for two samples, but only chromosome 1-9. One window per
#sample:
plotSample(data=sub.lymphoma,segments=list(uni.segments,multi.segments),sample=
    c(2,3),chrom=c(1:9),layout=c(3,3),seg.col=c("red","blue"),
    seg.lwd=c(3,2),onefile=FALSE)

#Zoom in on a particular region by setting xlim:
```

```
plotSample(data=sub.lymphoma,segments=uni.segments,sample=1,chrom=1,plot.ideo=
    FALSE,xlim=c(140,170))
```

`selectSegments` *Select multipcf segments*

Description

Selects multipcf segments based on a desired characteristic.

Usage

```
selectSegments(segments, what = "variance", thres = NULL, nseg = 10,
                 large = TRUE, p = 0.1)
```

Arguments

`segments`	a data frame containing segments found by `multipcf`.
`what`	the desired characteristic to base selection on. Must be one of "variance" (default),"length" and "aberration". See details below.
`thres`	an optional numeric threshold to be applied in the selection.
`nseg`	the desired number of segments to be selected, default is 10. Only used if `thres=NULL`.
`large`	logical value indicating whether segments with large (TRUE) or small (FALSE) variance, length or mean value should be selected when `what` is "variance", "length" or "aberration", respectively.
`p`	a number between 0 and 1 giving the minimum proportion of samples for which an aberration must be detected, default is 0.1. Only applicable if `what="aberration"`.

Details

The input in `what` determines how the segments are selected. Three options are available:

If `what="variance"` the variance of the segment values across all samples is calculated for each segment. If `thres` is specified, the subset of segments for which the variance is above (if `large=TRUE`) or below (if `large=FALSE`) the threshold is returned. If `thres` is not given by the user, a given number of segments determined by the input in `nseg` is selected; if `large=TRUE` this will be the `nseg` segments with the highest variance, whereas if `large=FALSE` the subset will consist of the `nseg` segments with the lowest variance.

If `what="length"` selection is based on the genomic length of the segments (end position minus start position). If `thres` is specified, the subset of segments for which the length is above (if `large=TRUE`) or below (if `large=FALSE`) this threshold is returned. If `thres` is left unspecified, a given number of segments determined by the input in `nseg` is selected; if `large=TRUE` this will be the `nseg` longest segments, whereas if `large=FALSE` it will be the `nseg` shortest segments.

If `what="aberration"` the aberration frequency is used to select the subset of segments. If `thres` is specified, the proportion of samples for which the segment value is above (if `large=TRUE`) or below (if `large=FALSE`) the threshold is calculated for each segment. The subset of segments where this frequency is above or equal to the proportion set by the parameter `p` is returned. If `thres` is not specified, the `nseg` segments with the highest (1-p)-quantile (if `large=TRUE`) or the lowest p-quantile (if `large=FALSE`) is returned.

Value

A list containing:

`sel.seg`	data frame containing the selected segments.

In addition, depending on the value of `what`:

`seg.var`	a vector giving the variance for each segment. Only returned if `what = "variance"`.
`seg.length`	a vector giving the length of each segment. Only returned if `what = "length"`.
`seg.ab.prop`	a vector giving the aberration proportion for each segment. Only returned if `what = "aberration"` and `thres` is specified.
`seg.quantile`	a vector giving the (1-p)- or p-quantile for each segment. Only returned if `what = "aberration"` and `thres=NULL`.

Author(s)

Gro Nilsen

See Also

`multipcf`

Examples

```
#Lymphoma data
data(lymphoma)

#Run multipcf
segments <- multipcf(lymphoma,gamma=12)

#Select the 10 segments with the highest variance:
sel.seg1 <- selectSegments(segments)

#Select the segments where the variance is below 0.001
sel.seg2 <- selectSegments(segments,thres=0.001,large=FALSE)

#Select the 5 longest segments:
sel.seg3 <- selectSegments(segments,what="length",nseg=5)

#Select the segments where 20 % of the samples have segment value of 0.2 or more:
sel.seg4 <- selectSegments(segments,what="aberration",thres=0.2,p=0.2)

#Select the 20 segments with the largest median:
sel.seg5 <- selectSegments(segments,what="aberration",nseg=20,p=0.5)
```

SNPdata *Artificial SNP array data*

Description

Artificial SNP array data containing a logR track and a BAF track

Usage

```
data(logR)
data(BAF)
```

Format

Two corresponding data sets containing 10000 probes with logR and BAF measurements, respectively, for 2 samples. The two first columns in both data sets contain chromosome numbers and local probe positions (in base pairs), while the subsequent columns contain logR-values and BAF-values in the two data sets, respectively.

Examples

```
#Get data
data(logR)
data(BAF)
```

---

subsetData *Retrieve a data subset*

Description

This function returns a subset of copy number data according to the input and the specified chromosomes and/or samples.

Usage

```
subsetData(data, chrom = NULL, sample = NULL, sep="\t", ...)
```

Arguments

data	either a data frame or the name of a tab-separated file from which copy number data can be read. The rows of the data frame or file should represent the probes. Column 1 must hold numeric or character chromosome numbers, column 2 the numeric local probe positions, and subsequent columns the numeric copy number measurements for one or more samples.
chrom	a numeric or character vector with chromosome(s) for which data should be selected. If unspecified, all chromosomes in data will be selected.
sample	a numeric vector indicating for which sample(s) data should be selected. The number(s) should correspond to the sample's place (in order of appearance) in data.
sep	the separator of the input files if data. Default is tab.
...	optional parameters to be passed to read.table in the case where data is to be read from file.

Value

A data frame containing the desired subset of data.

Author(s)

Gro Nilsen

Examples

```
#Load lymphoma data
data(lymphoma)

#Select data only for samples 1 and 6 and chromosomes 1:9:
sub.data <- subsetData(data=lymphoma,chrom=c(1:9),sample=c(1,6))
```

---

subsetSegments *Retrieve a subset of segments*

Description

This function returns a subset of segments according to the input and the specified chromosomes and/or samples.

Usage

```
subsetSegments(segments, chrom = NULL, sample = NULL, sep="\t", ...)
```

Arguments

segments	either a data frame or the name of a tab-separated file from which segmentation results can be read. Segmentation results may come from pcf, multipcf or aspcf.
chrom	a numeric or character vector with chromosome(s) for which segments should be selected. If unspecified, all chromosomes in segments will be selected.
sample	a numeric vector indicating for which sample(s) segments should be selected. The number(s) should correspond to the sample's place (in order of appearance) in segments.
sep	the separator of the input files if segments is a file. Default is tab.
...	optional parameters to be passed to read.table in the case where segments is to be read from file.

Value

A data frame containing the desired subset of segments.

Author(s)

Gro Nilsen

Examples

```
#Load lymphoma data
data(lymphoma)

#Select segments only for samples 1 and 6 and chromosomes 1:9:
segments <- pcf(lymphoma,gamma=12)
sub.segments <- subsetSegments(segments=segments,chrom=c(1:9),sample=c(1,6))
```

`winsorize` *Winsorization of copy number data*

Description

Outliers in copy number data are detected and modified using MAD or PCF Winsorization.

Usage

```
winsorize(data, pos.unit = "bp", arms = NULL, method = "mad", tau = 2.5,
          k = 25, gamma = 40, iter = 1, assembly = "hg19", digits = 4,
          return.outliers = FALSE, save.res = FALSE, file.names = NULL,
          verbose = TRUE)
```

Arguments

`data`	either a data frame or the name of a tab-separated file from which copy number data can be read. The rows of the data frame or file should represent the probes. Column 1 must hold numeric or character chromosome numbers, column 2 the numeric local probe positions, and subsequent column(s) the numeric copy number measurements for one or more samples. The header of copy number columns should give sample IDs.
`pos.unit`	the unit used to represent the probe positions. Allowed options are "mbp" (mega base pairs), "kbp" (kilo base pairs) or "bp" (base pairs). By default assumed to be "bp".
`arms`	optional character vector containing chromosome arms (denoted 'p' and 'q') corresponding to the chromosomes and positions found in `data`. If not specified chromosome arms are found using the built-in genome assembly version determined by `assembly`.
`method`	the Winsorization method to be applied, must be one of "mad" (default) or "pcf".
`tau`	Winsorization threshold, default is 2.5.
`k`	the half window size to be applied in median filtering, default is 25.
`gamma`	penalty for each discontinuity in the pcf curve, default is 40. Only applicable when `method="pcf"`.
`iter`	number of iterations in PCF Winsorization, default is 1.
`assembly`	a string specifying which genome assembly version should be applied to determine chromosome arms. Allowed options are "hg19", "hg18", "hg17" and "hg16" (corresponding to the four latest human genome annotations in the UCSC genome browser).

digits	the number of decimals to be applied when reporting results. Default is 4.
return.outliers	logical value indicating whether a data frame identifying outliers should be returned, default is FALSE.
save.res	logical value indicating whether results should be saved in text files, default is FALSE.
file.names	optional character vector of length two giving the name of the files where the Winsorized data and outlier statuses, respectively, should be saved if save.res=TRUE.
verbose	logical value indicating whether or not to print a progress message each time Winsorization is finished for a new chromosome arm.

Details

The copy number data are either MAD Winsorized or PCF Winsorized as described in Nilsen and Liestoel et al. (2012). Winsorization is done separately on each chromosome arm in each sample.

Value

If return.outliers = TRUE a list with the following components:

wins.data	a data frame with chromosome numbers in the first column, probe positions in the second and the Winsorized copy number values for the sample(s) in subsequent column(s).
wins.outliers	a data frame with chromosome numbers in the first column, probe positions in the second and outlier statuses for each sample in the subsequent column(s). The values +/- 1 indicate that the observation is an outlier, whereas the value 0 indicates that it is not.

If return.outliers = FALSE only the data frame containing the winsorized data is returned.

If save.res=TRUE the results are saved in text files with names as specified in file.names. If file.names=NULL, a folder named "Wins_res" is created in the working directory and Winsorized data and outlier statuses are saved in this directory in tab-separated files named wins.data.txt and wins.outliers.txt, respectively.

Note

Any missing values in data imply that the Winsorized value and outlier status for this probe will be missing as well. Also, if the number of probes within a chromosome arm is less than 2*k, Winsorization cannot be done and the data values are thus left unchanged.

Author(s)

Gro Nilsen, Knut Liestoel, Ole Christian Lingjaerde

References

Nilsen and Liestoel et al., "Copynumber: Efficient algorithms for single- and multi-track copy number segmentation", BMC Genomics 13:591 (2012), doi:10.1186/1471-2164-13-59

Examples

```
#Lymphoma data
data(lymphoma)
#Take out a smaller subset of 3 samples (using subsetData):
sub.lymphoma <- subsetData(lymphoma,sample=1:3)

#Do MAD Winsorization:
wins.data <- winsorize(data=sub.lymphoma)
```

Index