Package ‘aCGH’

May 13, 2024

Title Classes and functions for Array Comparative Genomic Hybridization data

Version 1.82.0

Date 2010-04-15

Depends R (>= 2.10), cluster, survival, multtest

Imports Biobase, grDevices, graphics, methods, stats, splines, utils

Author Jane Fridlyand <jfridlyand@cc.ucsf.edu>, Peter Dimitrov <dimitrov@stat.Berkeley.EDU>

Description Functions for reading aCGH data from image analysis output files and clone information files, creation of aCGH S3 objects for storing these data. Basic methods for accessing/replacing, subsetting, printing and plotting aCGH objects.

Maintainer Peter Dimitrov <dimitrov@stat.Berkeley.EDU>

License GPL-2

Collate aCGH.R aCGH.plotting.R aCGH.read.R aCGH.test.R clusterGenome.R funcs.dataplot.R heatmap.R hmm.R human.chrom.info.Jul03.R human.chrom.info.May04.R mergeLevels.R

LazyLoad yes

biocViews CopyNumberVariation, DataImport, Genetics

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

git_branch RELEASE_3_19

git_last_commit 617c195

git_last_commit_date 2024-04-30

Repository Bioconductor 3.19

Date/Publication 2024-05-13

Contents

`aCGH` *Class aCGH*

Description

Objects of this class represent batch of arrays of Comparative Genomic Hybridization data. In addition to that, there are slots for representing phenotype and various genomic events associated with aCGH experiments, such as transitions, amplifications, aberrations, and whole chromosomal gains and losses. Currently objects of class aCGH are represented as S3 classes which are named list of lists with functions for accessing elements of that list. In the future, it's anticipated that aCGH objects will be implemented using S4 classes and methods.

Details

One way of creating objects of class aCGH is to provide the two mandatory arguments to `create.aCGH` function: `log2.ratios` and `clones.info`. Alternatively aCGH object can be created using `aCGH.read.Sprocs` that reads Sproc data files and creates object of type aCGH.

Value

`log2.ratios` Data frame containing the log2 ratios of copy number changes; rows correspond to the clones and the columns to the samples (Mandatory).

`clones.info` Data frame containing information about the clones used for comparative genomic hybridization. The number of rows of `clones.info` has to match the number of rows in `log2.ratios` (Mandatory).

`phenotype` Data frame containing phenotypic information about samples used in the experiment generating the data. The number of rows of `phenotype` has to match the number of columns in `log2.ratios` (Optional).

`log2.ratios.imputed` Data frame containing the imputed log2 ratios. Calculate this using `impute.lowess` function; look at the examples below (Optional).

`hmm` The structure of the hmm element is described in `hmm`. Calculate this using `find.hmm.states` function; look at the examples below (Optional).

`hmm` Similar to the structure of the hmm element. Calculate this using `mergeHmmStates` function; look at the examples below (Optional).

`sd.samples` The structure of the sd.samples element is described in `computeSD.Samples`. Calculate this using `computeSD.Samples` function; look at the examples below (Optional). It is prerequisite that the hmm states are estimated first.

`genomic.events` The structure of the genomic.events element is described in `find.genomic.events`. Calculate this using `find.genomic.events` function; look also at the examples below. It is prerequisite that the hmm states and sd.samples are computed first. The `genomic.events` is used widely in variety of plotting functions such as `plotHmmStates`, `plotFreqStat`, and `plotSummaryProfile`.

`dim.aCGH` returns the dimensions of the aCGH object: number of clones by number of samples.

`num.clones` number of clones/number of rows of the log2.ratios data.frame.

`nrow.aCGH` same as `num.clones`.

`is.aCGH` tests if its argument is an object of class `aCGH`.

`num.samples` number of samples/number of columns of the log2.ratios data.frame.

`nrow.aCGH` same as `num.samples`.

`num.chromosomes` number of chromosomes processed and stored in the aCGH object.

`clone.names` returns the names of the clones stored in the clones.info slot of the aCGH object.

`row.names.aCGH` same as `clone.names`.

`sample.names` returns the names of the samples used to create the aCGH object. If the object is created using `aCGH.read.Sprocs`, these are the file names of the individual arrays.

`col.names.aCGH` same as `sample.names`.

`[.aCGH` subsetting function. Works the same way as `[.data.frame`.

Most of the functions/slots listed above have assignment operators '<-' associated with them.

Note

`clones.info` slot has to contain a list with at least 4 columns: Clone (clone name), Target (unique ID, e.g. Well ID), Chrom (chromosome number, X chromosome = 23 in human and 20 in mouse, Y chromosome = 24 in human and 21 in mouse) and kb (kb position on the chromosome).

Author(s)

Peter Dimitrov

See Also

`aCGH.read.Sprocs`, `find.hmm.states`, `computeSD.Samples`, `find.genomic.events`, `plotGenome`, `plotHmmStates`, `plotFreqStat`, `plotSummaryProfile`

Examples

```
## Creating aCGH object from log2.ratios and clone info files
## For alternative way look at aCGH.read.Sprocs help

datadir <- system.file(package = "aCGH")
datadir <- paste(datadir, "/examples", sep="")

clones.info <-
      read.table(file = file.path(datadir, "clones.info.ex.txt"),
                 header = TRUE, sep = "\t", quote="", comment.char="")
log2.ratios <-
      read.table(file = file.path(datadir, "log2.ratios.ex.txt"),
                 header = TRUE, sep = "\t", quote="", comment.char="")
pheno.type <-
      read.table(file = file.path(datadir, "pheno.type.ex.txt"),
                 header = TRUE, sep = "\t", quote="", comment.char="")
ex.acgh <- create.aCGH(log2.ratios, clones.info, pheno.type)

## Printing, summary and basic plotting for objects of class aCGH

data(colorectal)
colorectal
summary(colorectal)
sample.names(colorectal)
phenotype(colorectal)
plot(colorectal)

## Subsetting aCGH object

colorectal[1:1000, 1:30]

## Imputing the log2 ratios

log2.ratios.imputed(ex.acgh) <- impute.lowess(ex.acgh)

## Determining hmm states of the clones
## WARNING: Calculating the states takes some time
```

```
##in the interests of time, hmm-finding function is commented out
##instead the states previosuly save are assigned
##hmm(ex.acgh) <- find.hmm.states(ex.acgh)

hmm(ex.acgh) <- ex.acgh.hmm
hmm.merged(ex.acgh) <-
   mergeHmmStates(ex.acgh, model.use = 1, minDiff = .25)

## Calculating the standard deviations for each array

sd.samples(ex.acgh) <- computeSD.Samples(ex.acgh)

## Finding the genomic events associated with each sample

genomic.events(ex.acgh) <- find.genomic.events(ex.acgh)

## Plotting and printing the hmm states

plotHmmStates(ex.acgh, 1)
pdf("hmm.states.temp.pdf")
plotHmmStates(ex.acgh, 1)
dev.off()

## Plotting summary of the sample profiles

plotSummaryProfile(colorectal)
```

`aCGH.process` *Process data in aCGH object*

Description

This function takes object of class `aCGH`, and filters clones based on their mapping information and proportion missing. It also average duplicated clones and reports quality statistic.

Usage

```
aCGH.process(aCGH.obj, chrom.remove.threshold = 24,
                 prop.missing = 0.25, sample.quality.threshold = 0.4,
                 unmapScreen=TRUE, dupRemove = TRUE)
```

Arguments

`aCGH.obj`	Object of class aCGH

`chrom.remove.threshold`
: Chromosomes are ordered and numbered as usual, except for X and Y chromosome, which in for Homo sapiens genome have numbers 23 and 24 repsectivelly, in for Mus musculus 20 and 21, etc.

`prop.missing`
: Clones are screened out and if the proportion missing in the samples is `prop.missing` they are removed.

`sample.quality.threshold`
: Mark those samples that have their proportion of missing values `sample.quality.threshold`.

`unmapScreen`
: Indicator for whether clones with incomplete mapping information should be removed from the dataset. Note that leaving them in may cause plotting routines fail. Defaults to TRUE

`dupRemove`
: Indicator for whether clones with duplicate names should be averaged and removed from the dataset leaving only one occurence of each duplicated set.Defaults to TRUE

Value

Object of class `aCGH`.

Author(s)

Jane Fridlyand, Peter Dimitrov

See Also

`aCGH`

Examples

```
datadir <- system.file(package = "aCGH")
datadir <- paste(datadir, "/examples", sep="")

clones.info <-
      read.table(file = file.path(datadir, "clones.info.ex.txt"),
                 header = TRUE, sep = "\t", quote="", comment.char="")
log2.ratios <-
      read.table(file = file.path(datadir, "log2.ratios.ex.txt"),
                 header = TRUE, sep = "\t", quote="", comment.char="")
pheno.type <- read.table(file = file.path(datadir, "pheno.type.ex.txt"),header = TRUE, sep = "\t", quote="", comme
ex.acgh <- create.aCGH(log2.ratios, clones.info, pheno.type)
ex.acgh <-
    aCGH.process(ex.acgh, chrom.remove.threshold = 23, prop.missing = .25,
sample.quality.threshold = .4, unmapScreen=TRUE, dupRemove = FALSE)
ex.acgh
```

`aCGH.read.Sprocs` *Create object of class "aCGH" from Sproc files*

Description

This function reads in two-channel Array Comparative Genomic Hybridization Sproc files, flags them for bad quality and missing data, and creates object of class `aCGH`.

Usage

```
aCGH.read.Sprocs(fnames, latest.mapping.file = NULL, maxsd = 0.2,
                 minreplic = 2, chrom.remove.threshold = 24,
                 prop.missing = 0.25, sample.names = fnames,
                 sample.quality.threshold = 0.4,
                 cols = c("Log2Rat", "Log2StdDev", "NReplic", "Bad.P"),
                 unmapScreen=TRUE, dupRemove = TRUE)
```

Arguments

`fnames` a vector of character strings containing the file names of each Sproc data file.

`latest.mapping.file` The name of an optional file that contains newer clone mapping different from the clone mapping used at the time when the arrays were created.

`maxsd` maximum of standard deviation of log2 ratios used in pre-filtering.

`minreplic` minimum number of replicates per clone for a single chip used to calculate the log2 ratios.

`chrom.remove.threshold` Chromosomes are ordered and numbered as usual, except for X and Y chromosome, which in for Homo sapiens genome have numbers 23 and 24 repsectivelly, in for Mus musculus 20 and 21, etc.

`prop.missing` Clones are screened out and if the proportion missing in the samples is `prop.missing` they are removed.

`sample.names` Sample names. If they are missing, the file names are used after stripping the characters after the last dot in the filename if one exists; for example 'myfile.txt' becomes myfile.

`sample.quality.threshold` Mark those samples that have their proportion of missing values `sample.quality.threshold`.

`cols` character vector of length 4 containing the following Sproc file column names: log2 ratios, std. deviations of the log2 ratios, number of replicates for each clone and flags for bad clones. Defaults to c("Log2Rat", "Log2StdDev", "NReplic", "Bad.P"). Note that all the whitespace characters in the column names will be replaced with dots.

`unmapScreen` Indicator for whether clones with incomplete mapping information should be removed from the dataset. Note that leaving them in may cause plotting routines fail. Defaults to TRUE

dupRemove Indicator for whether clones with duplicate names should be averaged and removed from the dataset leaving only one occurence of each duplicated set.Defaults to TRUE

Value

Object of class aCGH.

Author(s)

Jane Fridlyand, Peter Dimitrov

See Also

aCGH

Examples

```
datadir <- system.file("examples", package = "aCGH")
latest.mapping.file <-
      file.path(datadir, "human.clones.info.Jul03.txt")
ex.acgh <-
aCGH.read.Sprocs(dir(path = datadir,pattern = "sproc",
full.names = TRUE), latest.mapping.file,
chrom.remove.threshold = 23)
ex.acgh

## Testing if creating the object went right. Should all be true.

all(log2.ratios(ex.acgh)[ 1, ] == c(-0.077698 , 0.007389))
clone.name <- "HumArray2H10_T30"
all(log2.ratios(ex.acgh)[ clone.name, ] == c(0.025567 , -0.036908))
```

---

aCGH.test *Testing association of aCGH clones with censored or continuous outcomes*

Description

aCGH.test function tests for association of each clone in an univariate manner with censored or continous outcome by fitting Cox proportional hazards model or linear regression model. There is also an alternative to Cox prop. hazards - testing for differences in survival curves defined by the groups in the outcome variable using the G^{ρ} family of tests.

Usage

```
aCGH.test(aCGH.obj, rsp, test = c("survdiff","coxph",
          "linear.regression"), p.adjust.method = "fdr",imputed=TRUE,
           subset = NULL, strt = NULL, ...)
```

Arguments

`aCGH.obj`	aCGH object containing clones' log2 ratios.
`rsp`	Response variable which is either `Surv` object from `survival` package or continous outcome.
`test`	Currently only three values are allowed - "coxph", "survdiff", and "linear.regression", which test for association using Cox proportional hazards model, G^ρ family of tests (`survdiff`) or linear model.
`p.adjust.method`	This is a parameter controlling how the p-values from the univariate tests are going to be adjusted for multiple testing. Default value is Benjamini & Hochberg (1995) FDR method. Please refer to `p.adjust` function for more help.
`imputed`	Whether imputed or original log2ratios should be used. Default is TRUE (imputed).
`subset`	Specifies subset index of clones to be tested.
`strt`	Aptional strata variable for splitting the data in different strata.
`...`	Optional parameters passed further along to each of the univariate testing functions.

Value

A data frame similar to the result returned from `mt.maxT` function from `multtest` package with components:

`index`	Vector of row indices, between 1 and `nrow(X)`, where rows are sorted first according to their adjusted p-values, next their unadjusted p-values, and finally their test statistics.
`teststat`	Vector of test statistics, ordered according to `index`. To get the test statistics in the original data order, use `teststat[order(index)]`.
`rawp`	Vector of raw (unadjusted) p-values, ordered according to `index`.
`adjp`	Vector of adjusted p-values, ordered according to `index`.

Author(s)

Peter Dimitrov

See Also

`aCGH`, `Surv`, `mt.maxT`, `coxph`, `survdiff`, `p.adjust`

clusterGenome *clustering and heatmap*

Description

This function clusters samples and shows their heatmap

Usage

```
clusterGenome(aCGH.obj,
                    response = as.factor(rep("All", ncol(aCGH.obj))),
                    chrominfo = human.chrom.info.Jul03, cutoff=1,
                    lowCol = "red", highCol = "green", midCol = "black",
                    ncolors = 50, byclass = FALSE, showaber = FALSE,
                    amplif = 1, homdel = -0.75,
                    samplenames = sample.names(aCGH.obj),
                    vecchrom = 1:23, titles = "Image Plot",
                    methodS = "ward.D", dendPlot = TRUE, imp = TRUE,
                    categoricalPheno = TRUE)
```

Arguments

aCGH.obj	object of class aCGH here
response	phenotype of interest. defaults to the same phenotype assigned to all samples
chrominfo	a chromosomal information associated with the mapping of the data
cutoff	maximum absolute value. all the values are floored to +/-cutoff depending on whether they are positive of negative. defaults to 1
ncolors	number of colors in the grid. input to maPalette. defaults to 50
lowCol	color for the low (negative) values. input to maPalette. defaults to "red"
highCol	color for the high (positive) values. input to maPalette. defaults to "green"
midCol	color for the values close to 0. input to maPalette. defaults to "black"
byclass	logical indicating whether samples should be clustered within each level of the phenotype or overall. defaults to F
showaber	logical indicating whether high level amplifications and homozygous deletions should be indicated on the plot. defaults to F
amplif	positive value that all observations equal or exceeding it are marked by yellow dots indicating high-level changes. defaults to 1
homdel	negative value that all observations equal or below it are marked by light blue dots indicating homozygous deletions. defaults to -0.75
samplenames	sample names
vecchrom	vector of chromosomal indeces to use for clustering and to display. defaults to 1:23
titles	plot title. defaults to "Image Plots"

`methodS` clustering method to cluster samples. defaults to "ward.D"

`dendPlot` logical indicating whether dendogram needs to be drawn. defaults to T.

`imp` logical indicating whether imputed or original values should be used. defaults to T, i.e. imputed.

`categoricalPheno` logical indicating whether phenotype is categorical. Continious phenotypes are treated as "no groups" except that their values are dispalyed.defaults to TRUE.

Details

This functions is a more flexible version of the `heatmap`. It can cluster within levels of categorical phenotype as well as all of the samples while displaying phenotype levels in different colors. It also uses any combination of chromosomes that is requested and clusters samples based on these chromosomes only. It draws the chromosomal boundaries and displays high level changes and homozygous deletions. If phenotype if not categical, its values may still be displayed but groups are not formed and byclass = F. Image plot has the samples reordered according to clustering order.

See Also

`aCGH` `heatmap`

Examples

```
data(colorectal)

#cluster all samples using imputed data on all chromosomes (autosomes and X):

clusterGenome(colorectal)

#cluster samples within sex groups based on 3 chromosomes individually.
#use non-imputed data and  do not show dendogram. Indicate amplifications and
#homozygous deletions.

clusterGenome(colorectal, response = phenotype(colorectal)$sex,
                   byclass = TRUE, showaber = TRUE, vecchrom = c(4,8,9),
                   dendPlot = FALSE, imp = FALSE)

#cluster samples based on each chromosome individualy and display age. Show
#gains in red and losses in green. Show aberrations and use values < -1
#to identify homozgous deletions. Do not show dendogram.

pdf("plotimages.pdf", width = 11, height = 8.5)
for (i in 1:23)
    clusterGenome(colorectal,
                        response = phenotype(colorectal)$age,
                        chrominfo = human.chrom.info.Jul03,
                        cutoff = 1, ncolors = 50, lowCol="green",
                        highCol="red", midCol="black", byclass = FALSE,
                        showaber = TRUE, homdel = -1, vecchrom = i,
                        titles = "Image Plot", methodS = "ward.D",
                        dendPlot = FALSE, categoricalPheno = FALSE)
```

```
dev.off()
```

`colorectal` *Colorectal array CGH dataset*

Description

The colorectal dataset is an object of class `aCGH`. It represents a collection of 124 array CGH profiles of primary colorectal tumors and their derived attributes. Each sample was measured on the BAC clone DNA microarray with approximate resolution of 1.4 Mb per clone. There were approximately 2400 clones spotted on the array and each clone was printed in triplicates located immediately next to each other. Each array consisted of the 16 (4 by 4) subarrays. The clones were mapped on the July 03 UCSC freeze. There were a number of the discrete and continious phenotypes associated with the samples such as age, mutation status for various markers, stage, location and so on. All images were quantified and normalized by Dr. Taku Tokuyasu using custom image software SPOT and postprocessing custom software SPROC.

Usage

```
data(colorectal)
```

Source

These data were generated at Dr. Fred Waldman's lab at UCSF Cancer Center by K. Nakao and K. Mehta. The manuscript describing the data and the analysis are described in High-resolution analysis of DNA copy number alterations in colorectal cancer by array-based comparative genomic hybridization, *Carcinogenesis*, 2004, Nakao et. al.

References

Nakao et. al., High-resolution analysis of DNA copy number alterations in colorectal cancer by array-based comparative genomic hybridization, *Carcinogenesis*, 2004 Jain et. al, Fully automatic quantification of microarray image data, *Genome Research*, 2003

See Also

`aCGH` `plotGenome`

Examples

```
data(colorectal)
## WARNING: plotting the heatmap takes some time
plot(colorectal)
plotGenome(colorectal[,1:2])
```

computeSD.func *Function to estimate experimental variability of a sample*

Description

This functions estimate experimental variability of a given sample. This value can be used to rank samples in terms of the quality as well as to derive thresholds for declaring gained and lost clones.

Usage

```
computeSD.Samples(aCGH.obj, maxChrom = 22, maxmadUse = .3,
                  maxmedUse = .5, maxState = 3, maxStateChange = 100,
  minClone = 20)
computeSD.func(statesres, maxmadUse = 0.2, maxmedUse = 0.2,
               maxState = 3, maxStateChange = 100, minClone = 20,
               maxChrom = 22)
```

Arguments

aCGH.obj	Object of class aCGH.
statesres	The states.hmm object, generally is the output of mergeFunc.
maxmadUse	Maximum median absolute deviation allowed to controbute to the overall variability calculation.
maxmedUse	Maximum median value for a state allowed to contribute to the calculation.
maxState	Maximum number of the states on a given chromosome for the states from that chromosome to be allowed to enter noise variability calculation.
maxStateChange	Maximum number of changes from state to state on a given chromosome for that chromosome to enter noise variability calculation.
minClone	Minimum number of clones in a state for clones in that sate to enter variability calculation.
maxChrom	Maxiumum chromosomal index (generally only autosomes are used for this calculation.

Details

Median absolute deviation is estimated in all the states passing the criteria defined by the parameters of the function. Then median of all MADs on individual chromosomes as well as across all chromosomes is taken to estimate chromosomal experimental variability and sample experimental variability.

Value

madChrom	Returns a matrix containing estimated variability for each chromosome for each sample.
madGenome	Returns a vector with estimate of experimental varibility for each sample.

Author(s)

Jane Fridlyand

References

Application of Hidden Markov Models to the analysis of the array CGH data, Fridlyand et.al., *JMVA*, 2004

See Also

aCGH

---

fga.func — *Function to compute fraction of genome altered for each sample*

Description

This function outputs lists containing proportions of the genome that are gained and lost for each sample.

Usage

```
fga.func(aCGH.obj, thres = 0.25, factor = 2.5,
         samplenames = sample.names(aCGH.obj),
         chrominfo = human.chrom.info.Jul03)
```

Arguments

aCGH.obj	An object of aCGH class
thres	either a vector providing unique threshold for each sample or a vector of the same length as number of samples providing sample-specific threshold. If 'aCGH.obj' has non-null 'sd.samples', then threshold is automatically replaced by tumor-specific sd multiplied by 'factor'. Clone is considered to be gained if it is above the threshold and lost if it is below negative threshold. Defaults to 0.25
factor	specifies the number by which experimental variability should be multiples. Used only when tumor specific variability in 'aCGH.obj' is not NULL or when factor is greater than 0. Defaults to 2.5.
samplenames	Sample names. Default is sample.names(aCGH.obj)
chrominfo	A chromosomal information associated with the mapping of the data. Default is human.chrom.info.Jul03 data frame

Value

gainP	Vector of proportion of genome gained for each sample
lossP	Vector of proportion of genome lost for each sample

Author(s)

Jane Fridlyand, Ritu Roydasgupta

Examples

```
data(colorectal)

col.fga <- fga.func(colorectal, factor=3,chrominfo=human.chrom.info.Jul03)
cbind(gainP=col.fga$gainP,lossP=col.fga$lossP)[1:5,]
```

---

find.genomic.events — *Finds the genomic events associated with each of the array CGH samples*

Description

Finds the genomic events associated with each of the array CGH samples. Events include whole chromosomal gains and losses, aberrations, transitions, amplifications and their respective counts and sizes. The hmm states has to be computed before using this function.

Usage

```
find.genomic.events(aCGH.obj, maxChrom = 23, factor = 5, maxClones = 1,
                    maxLen = 1000, absValSingle = 1, absValRegion = 1,
                    diffVal1 = 1, diffVal2 = .5, maxSize = 10000,
                    pChrom.min = .9, medChrom.min = .1)
```

Arguments

aCGH.obj	Object of class aCGH.
maxChrom	Highest chromosomal number to find events.
factor	Determines outliers. See findOutliers.func.
maxClones	Determines aberrations. See findAber.func.
maxLen	Determines aberrations. See findAber.func.
absValSingle	Determines amplifications. See findAmplif.func.
absValRegion	Determines amplifications. See findAmplif.func.
diffVal1	Determines amplifications. See findAmplif.func.
diffVal2	Determines amplifications. See findAmplif.func.
maxSize	Determines amplifications. See findAmplif.func.
pChrom.min	Determines whole chromosomal gains and losses. Chromosome should contain no transitions, have its absolute median equal or greater than medChrom.min and at least medChrom.min has to be greater or less than 0.
medChrom.min	Determines whole chromosomal gains and losses. Chromosome should contain no transitions, have its absolute median equal or greater than medChrom.min and at least medChrom.min has to be greater or less than 0.

Details

The default parameters generally work. Threshold for merging may be changed depending on the expected normal cell contamination and/or expected magnitude of the changes. AIC model generally works, however, may need to be readjusted depending on how liberal or conservative one wants to be in finding genomic events. We recommend BIC criterion with delta = 1 for noisier data.

Value

`num.transitions`
: matrix of dimensions maxChrom by number of samples. It contains number of transitions that were recorded on a given chromosome for a given sample.

`num.amplifications`
: matrix of dimensions maxChrom by number of samples It contains number of amplifications that were recorded on a given chromosome for a given sample.

`num.aberrations`
: matrix of dimensions maxChrom by number of samples. It contains number of focal aberrations that were recorded on a given chromosome for a given sample.

`num.outliers`
: matrix of dimensions maxChrom by number of samples. It contains number of outliers that were recorded on a given chromosome for a given sample.

`num.transitions.binary`
: binary matrix of dimensions maxChrom by number of samples. Non-zero entry indicates whether 1 or more transitions were recorded on a given chromosome for a given sample.

`num.amplifications.binary`
: binary matrix of dimensions maxChrom by number of samples. Non-zero entry indicates whether 1 or more amplifications were recorded on a given chromosome for a given sample.

`num.aberrations.binary`
: binary matrix of dimensions maxChrom by number of samples. Non-zero entry indicates whether 1 or more focal aberrations were recorded on a given chromosome for a given sample.

`num.outliers.binary`
: binary matrix of dimensions maxChrom by number of samples. Non-zero entry indicates whether 1 or more outliers were recorded on a given chromosome for a given sample.

`whole.chrom.gain.loss`
: matrix of dimensions maxChrom by number of samples. Positive entry indicates that a given chromosome was gained in a given sample, negative entry indicates that a given chromosome was lost in a given sample, 0 entry is normal chromosome and NA marks chromosomes with one or more transition.

`size.amplicons`
: matrix of dimensions maxChrom by number of samples. Reports size of a given chromosome that is amplified (kb units) in a given sample.

`num.amplicons`
: matrix of dimensions maxChrom by number of samples. Reports number of disjoint amplicons on a given chromosome for a given sample.

`outliers`
: list containing 3 matrices of dimensions number of clones by number of samples. See `findOutliers.func`.

aberrations	list containing a matrix of dimensions number of clones by number of samples. See findAber.func.
transitions	list containing 2 matrices of dimensions number of clones by number of samples. See findTrans.func.
amplifications	list containing a matrix of dimensions number of clones by number of samples. See findAmplif.func.

See Also

aCGH find.hmm.states mergeFunc findAber.func findTrans.func findAmplif.func findOutliers.func

---

find.hmm.states *Determines states of the clones*

Description

This function runs unsupervised HMM algorithm and produces the essentual state information which is used for the subsequent structure determination.

Usage

```
hmm.run.func(dat, datainfo = clones.info, vr = 0.01, maxiter = 100,
             aic = TRUE, bic = TRUE, delta = NA, eps = 0.01)
find.hmm.states(aCGH.obj, ...)
```

Arguments

aCGH.obj	object of class aCGH.
dat	dataframe with clones in the rows and samples in the columns
datainfo	dataframe containing the clones information that is used to map each clone of the array to a position on the genome. Has to contain columns with names Clone/Chrom/kb containing clone names, chromosomal assignment and kb positions respectively
vr	Initial experimental variance
maxiter	Maximum number of iterations
aic	TRUE or FALSE variable indicating whether or nor AIC criterion should be used for model selection (see DETAILS)
bic	TRUE or FALSE variable indicating whether or nor BIC criterion should be used for model selection (see DETAILS)
delta	numeric vector of penalty factors to use with BIC criterion. If BIC is true, delta=1 is always calculated (see DETAILS)
eps	parameter controlling the convergence of the EM algorithm.
...	All the parameters that can be passed to find.hmm.states except dat and datainfo.

Details

One or more model selection criterion is used to determine number of states on each chromosomes. If several are specified, then a separate matrix is produced for each criterion used. Delta is a fudge factor in BIC criterion: $\delta BIC(\gamma) = \log RSS(\gamma) + q_\gamma \delta \log n/n$. Note that delta = NA leads to conventional BIC. (Broman KW, Speed TP (2002) A model selection approach for the identification of quantitative trait loci in experimental crosses (with discussion). J Roy Stat Soc B 64:641-656, 731-775)

find.hmm.states(aCGH.obj, ...) uses aCGH object instead of log2 ratios matrix dat. Equivalent representation (assuming normally distributed residuals) is to write -loglik(gamma) = n/2*log(RSS)(gamma) and then bic= -loglik+log(n)*k*delta/2 and aic = -loglik+2*k/2

Value

Two lists of lists are returned. Each list contains information on the states with each of the specified model selection criteria. E.g., if AIC = T, BIC = T and delta = c(1.5), then each list will contain three lists corresponding to AIC, BIC(1) and BIC(1.5) as the 1st,2nd and 3rd lists repsectively. If AIC is used, it always comes first followed by BIC and then deltaBIC in the order of delta vector.

`states.hmm` Each of the sublists contains 2+ 6*n columns where the first two columns contain chromosome and kb positions for each clone in the dataset supplied followed up by 6 columns for each sample where n = number of samples.

column 1 = state

column 2 = smoothed value for a clone

column 3 = probability of being in a state

column 4 = predicted value of a state

column 5 = dispersion

column 6 = observed value

`nstates.hmm` Each of the sublists contains a matrix with each row corresponding to a chromosome and each column to a sample. The entries indicate how many different states were identified for a given sample on a given chromosome

WARNING

When algortihm fails to fit an HMM for a given number of states on a chromosome, it prints a warning.

Author(s)

Jane Fridlyand

References

Application of Hidden Markov Models to the analysis of the array CGH data, Fridlyand et.al., *JMVA*, 2004

See Also

aCGH

Examples

```
datadir <- system.file("examples", package = "aCGH")
latest.mapping.file <-
      file.path(datadir, "human.clones.info.Jul03.txt")
ex.acgh <-
    aCGH.read.Sprocs(dir(path = datadir,pattern = "sproc",
                     full.names = TRUE), latest.mapping.file,
                     chrom.remove.threshold = 23)
ex.acgh

data(colorectal)
#in the interests of time, we comment the actual hmm-finding function out.
#hmm(ex.acgh) <- find.hmm.states(ex.acgh, aic = TRUE, delta = 1.5)
summary(ex.acgh)
```

`findAber.func` *Function to determines focal aberrations*

Description

The function identifies clones that are focal aberrations.

Usage

```
findAber.func(maxClones = 1, maxLen = 1000, statesres)
```

Arguments

`maxClones`	Maximum number of clones assigned to the same state which can be considered to be focal aberrations
`maxLen`	Maximum lengeth of the region containing clones assigned to the state so that those clones can be considered to be focal aberrations
`statesres`	The states output of the `hmm.run.func`

Details

The focal aberrations are the one or more clones assigned to the state different from the states of the surrounding clones. They may indicate copy number polymorphisms or interesting high or low focal changes.

Value

`aber`	Binary matrix with a row for each clone and column for each sample. 1 indicates presence of a focal aberrations, 0 lack of such.

Author(s)

Jane Fridlyand

References

"Application of Hidden Markov Models to the analysis of the array CGH data", Fridlyand et.al., JMVA, 2004

`findAmplif.func` *Function to determine high level amplifications*

Description

This function identifies high level amplifications by considering the height, the width of an amplicon relative to the urrounding clones. Only narrow peaks much higher than its neigbors are considered as high level amplifications.

Usage

```
findAmplif.func(absValSingle = 1, absValRegion = 1.5, diffVal1 = 1,
diffVal2 = 0.5, maxSize = 10000, translen.matr, trans.matr, aber,
outliers, pred, pred.obs, statesres)
```

Arguments

`absValSingle`	A clone is declared to be an amplification if it is a focal aberration or an outlier and its value exceeds absValSingle
`absValRegion`	A clone is an amplification if if a clone belong to a region with width less than maxSize and observed value for a clones is greater than absValRegion
`diffVal1`	Clone is an amplification if it is an aberration and greater by diffVal1 than max of the two surrounding stretches
`diffVal2`	Clone is an amplification if it is an outlier, its observed values is greater by diffVal2 than max of the two surrounding stretches
`maxSize`	The clones may not be declared as amplifications if they belong to the states with spanning more than maxSize
`translen.matr`	State length matrix. The output of the `findTrans.func`
`trans.matr`	Transition matrix. The output of the `findTrans.func`
`aber`	Aberration matrix. The output of the `findAber.func`
`outliers`	Outliers matrix. The output of the `findOutliers.func`
`pred`	Predicted values matrix. The output of the `findOutliers.func`
`pred.obs`	Predicted values matrix with observed values assigned to the outliers. The output of the `findOutliers.func`
`statesres`	The states output of the `hmm.run.func`

Details

Note that all the distances are in Megabases and all the heights are on log2ratio scale.

Value

`amplif.matrix` Binary matrix with a row for each clone and column for each sample. "1" indicates amplification

...

Author(s)

Jane Fridlyand

References

Application of Hidden Markov Models to the analysis of the array CGH data, Fridlyand et.al., *JMVA*, 2004

See Also

`aCGH`

`findOutliers.func` *Function to identify outlier clones*

Description

The function identified the clones that are outliers.

Usage

```
findOutliers.func(thres, factor = 4, statesres)
```

Arguments

`thres`	Estimate of experimental variability, generally, madGenome
`factor`	Factor indicating how many standard
`statesres`	The states output of the `hmm.run.func`

Details

The outliers are the clones that are dissimilar enough from the clones assigned to the same state. Magnitude of the factor determines how many MADs away from a median a value needs to be to be declared an outlier. Outliers consitent over many samples may indicate technological artificat with that clone or possibly copy number polymorpism.

Value

`outlier`	Binary matrix with a row for each clone and column for each sample. "1" indicates outlier, 0 otherwise.
`pred.obs.out`	Matrix with a row for each clone and column for each sample. The entries are the median value for the state with outliers exceluded for all clones but outliers. The value if the observed value for the outliers.
`pred.out`	Matrix with a row for each clone and column for each sample. The entries are the median value for the state

Author(s)

Jane Fridlyand

References

Application of Hidden Markov Models to the analysis of the array CGH data, Fridlyand et.al., *JMVA*, 2004

See Also

`aCGH`

`findTrans.func` *Funtion identifying the transitions*

Description

This function identifies the start and end of the states (regions with the constant estimated copy number).

Usage

```
findTrans.func(outliers, aber, statesres)
```

Arguments

`outliers`	Binary matrix of the outliers (generally output of the `findOutliers.func`)
`aber`	Binary matrix of the focal aberrations (generally output of the `findAber.func`)
`statesres`	The states output of the `hmm.run.func`

Details

The transitions end is placed at the last non-focal aberration clone of the contiguous region containing clones belonging to the same state and transitions start is placed at the first non-focal aberration clone of the contiguous region containing clones belonging to the same state.

Value

`trans.matrix` Matrix with a row for each clone and column for each sample. The starts of the states are indicated by "1" , the end are by "2" and the focal aberrations are coded as "3"

`translen.matrix` Matrix with a row for each clone and column for each sample. The entries are the length of the region to which a clone belongs. Zero length is assigned to the focal aberrations. This output may be buggy at the moment.

Author(s)

Jane Fridlyand

References

Application of Hidden Markov Models to the analysis of the array CGH data, Fridlyand et.al., *JMVA*, 2004.

See Also

aCGH

---

gainLoss — *Function to compute proportion of gains and losses for each clones*

Description

This function outputs lists containing proportion of gains and losses for each clone.

Usage

```
gainLoss(dat, cols, thres=0.25)
```

Arguments

`dat`	log2ratios of the relevant array CGH object
`cols`	indeces of the samples to use
`thres`	global or tumor-specific threshold. defaults to 0.25

Value

`gainP`	Vector of proportion gained for each clones
`lossP`	Vector of proportion lost for each clones

Author(s)

Jane Fridlyand

See Also

plotFreqStat

Examples

```
data(colorectal)

## Use mt.maxT function from multtest package to test
## differences in group means for each clone grouped by sex
##use only clones with show gain or loss in at least 10% of the samples
colnames(phenotype(colorectal))
sex <- phenotype(colorectal)$sex
sex.na <- !is.na(sex)
colorectal.na <- colorectal[ ,sex.na, keep = TRUE ]
factor <- 2.5
minChanged <- 0.1
gainloss <- gainLoss(log2.ratios(colorectal.na), cols=1:ncol(colorectal.na), thres=factor*sd.samples(colorectal.
ind.clones.use <- which(gainloss$gainP >= minChanged | gainloss$lossP>= minChanged)
#create filtered dataset
colorectal.na <- colorectal.na[ind.clones.use,keep=TRUE]
dat <- log2.ratios.imputed(colorectal.na)
resT.sex <- mt.maxT(dat, sex[sex.na],test = "t.equalvar", B = 1000)


## Plot the result along the genome
plotFreqStat(colorectal.na, resT.sex, sex[sex.na],factor=factor,titles = c("Male", "Female"))
```

---

heatmap *Creates heatmap array CGH objects*

Description

Clusters samples and produces heatmapp of the observed log2ratios.

Usage

```
heatmap(x, imp = TRUE, Rowv = NA, Colv = NULL, distfun = dist,
        hclustfun = hclust, add.expr, symm = FALSE,
        revC = identical(Colv, "Rowv"), scale = "none",
        na.rm = TRUE, margins = c(5, 5), ColSideColors,
        RowSideColors, cexRow = 0.2 + 1 / log10(nr),
        cexCol = 0.2 + 1 / log10(nc), labRow = NULL,
        labCol = NULL, main = NULL, xlab = NULL, ylab = NULL,
        verbose = getOption("verbose"), methodR = "ward.D",
        methodC = "ward.D", zlm = c(-0.5, 0.5), ...)
```

Arguments

`x`	object of the `aCGH` object
`imp`	logical variable indicating whether `log2.ratios.imputed` or `log2.ratios` slot of `aCGH` should be used. Defaults to imputed value (TRUE).
`Rowv`	determines if and how the row dendrogram should be computed and reordered. Either a 'dendrogram' or a vector of values used to reorder the row dendrogram or 'NA' to suppress any row dendrogram (and reordering) or by default, 'NULL'
`Colv`	determines if and how the column dendrogram should be reordered. Has the same options as the `Rowv` argument above and additionally when `x` is a square matrix, `Colv = "Rowv"` means that columns should be treated identically to the rows.
`distfun`	function used to compute the distance (dissimilarity) between both rows and columns. Defaults to 'dist'.
`hclustfun`	function used to compute the hierarchical clustering when 'Rowv' or 'Colv' are not dendrograms. Defaults to 'hclust'
`add.expr`	expression that will be evaluated after the call to 'image'. Can be used to add components to the plot.
`symm`	logical indicating if 'x' should be treated *symm*etrically; can only be true when 'x' is a square matrix.
`revC`	logical indicating if the column order should be 'rev'ersed for plotting, such that e.g., for the symmetric case, the symmetry axis is as usual.
`scale`	character indicating if the values should be centered and scaled in either the row direction or the column direction, or none. The default is "row" if `symm` false, and "none" otherwise.
`na.rm`	logical indicating whether 'NA"s should be removed.
`margins`	numeric vector of length 2 containing the margins (see 'par(mar= *)') for column and row names, respectively.
`ColSideColors`	(optional) character vector of length 'ncol(x)' containing the color names for a horizontal side bar that may be used to annotate the columns of 'x'.
`RowSideColors`	(optional) character vector of length 'nrow(x)' containing the color names for a vertical side bar that may be used to annotate the rows of 'x'.
`cexRow`, `cexCol`	positive numbers, used as 'cex.axis' in for the row or column axis labeling. The defaults currently only use number of rows or columns, respectively.
`labRow`, `labCol`	character vectors with row and column labels to use; these default to 'rownames(x)' or 'colnames(x)', respectively.
`main`, `xlab`, `ylab`	main, x- and y-axis titles;
`verbose`	logical indicating if information should be printed.
`methodR`	method to use for clustering rows. defaults to "ward.D"
`methodC`	method to use for clustering columns. defaults to "ward.D"
`zlm`	all the values greater or equal than zlm are set to zlm - 0.01. a;; value less or equal to -zlm are set to -zlm + 0.01
`...`	additional arguments passed on to 'image', e.g., 'col' specifying the colors.

Details

This function is almost identical to the `heatmap` in base R. The slight modifications are that (1) a user can specify clustering method for rows and columns; (2) all the values outside specified limits are floored to be 0.01 less than a limit; (3) default values are different. Note that using default option of `imp` (TRUE) produces nicer looking plots as all missing values are removed.

Value

Invisibly, a list with components

`crowInd`	row index permutation vector as returned by `order.dendrogram`
`colInd`	row index permutation vector as returned by `order.dendrogram`

References

heatmap function in base R

See Also

`aCGH` `clusterGenome`

Examples

```
#default plotting method for the aCGH object
data(colorectal)
plot(colorectal)

#to produce smoother looking heatmap, use imp = T: this will use imputed
#slot of aCGH object

plot(colorectal, imp = TRUE)
```

---

human.chrom.info.Jul03

Basic Chromosomal Information for UCSC Human Genome Assembly July 2003 freeze

---

Description

This dataset contains basic chromosomal information for UCSC Human Genome Assembly July 2003 freeze. human.chrom.info.Jul03 is loaded automatically with the aCGH package.

Usage

```
human.chrom.info.Jul03
```

Format

A data frame with 24 observations on the following 3 variables.

chrom Chromosomal index, X is coded as 23 and Y as 24.

length Length of each chromosome in kilobases.

centromere Location of the centromere on the chromosome (kb).

Details

This file is used for many plotting functions and needs to correspond to `clones.info` mapping file. The centromeric location is approximately extimated by taking mid-point between the last fish-mapped clone on the p-arm and the first fish-mapped clone on the q-arm using relevant UCSC freeze. For an alternative freeze, one needs to manually create a 3-column file of the format described above.

Source

http://genome.ucsc.edu/cgi-bin/hgText

`human.chrom.info.May04` *Basic Chromosomal Information for UCSC Human Genome Assembly May 2004 freeze*

Description

This dataset contains basic chromosomal information for UCSC Human Genome Assembly May 2004 freeze. human.chrom.info.May04 is loaded automatically with the aCGH package.

Usage

```
human.chrom.info.May04
```

Format

A data frame with 24 observations on the following 3 variables.

chrom Chromosomal index, X is coded as 23 and Y as 24.

length Length of each chromosome in kilobases.

centromere Location of the centromere on the chromosome (kb).

Details

This file is used for many plotting functions and needs to correspond to `clones.info` mapping file. The centromeric location is approximately extimated by taking mid-point between the last fish-mapped clone on the p-arm and the first fish-mapped clone on the q-arm using relevant UCSC freeze. For an alternative freeze, one needs to manually create a 3-column file of the format described above.

Source

http://genome.ucsc.edu/cgi-bin/hgText

`impute.HMM` *Imputing log2 ratios using HMM*

Description

Imputing log2 ratios using the output of the HMM segmenttation

Usage

```
impute.HMM(aCGH.obj, chrominfo = human.chrom.info.Jul03, maxChrom =
23, use.BIC = TRUE)
```

Arguments

`aCGH.obj`	Object of class aCGH.
`chrominfo`	a chromosomal information associated with the mapping of the data
`maxChrom`	Highest chromosome to impute.
`use.BIC`	logical parameter; if true impute missing values based on the Hidden Markov Model selected using Bayesian Information Criterion impute missing data, otherwise use AIC.

Details

See details in `aCGH` discussion.

Value

Computes and returns the imputed log2 ratio matrix of the aCGH object using the output of the Hidden Markov Model segmentation done by invoking `find.hmm.states` function.

See Also

`aCGH`, `find.hmm.states`, `impute.lowess`.

Examples

```
datadir <- system.file(package = "aCGH")
datadir <- paste(datadir, "/examples", sep="")

clones.info <-
      read.table(file = file.path(datadir, "clones.info.ex.txt"),
                 header = TRUE, sep = "\t", quote="", comment.char="")
log2.ratios <-
      read.table(file = file.path(datadir, "log2.ratios.ex.txt"),
```

```
                  header = TRUE, sep = "\t", quote="", comment.char="")
ex.acgh <- create.aCGH(log2.ratios, clones.info)

## Imputing the log2 ratios

hmm(ex.acgh) <- find.hmm.states(ex.acgh, aic = TRUE, delta = 1.5)
log2.ratios.imputed(ex.acgh) <- impute.HMM(ex.acgh)
```

`impute.lowess` *Imputing log2 ratios*

Description

Imputing log2 ratios

Usage

```
impute.lowess(aCGH.obj, chrominfo = human.chrom.info.Jul03, maxChrom =
23, smooth = 0.1)
```

Arguments

`aCGH.obj`	Object of class aCGH.
`chrominfo`	a chromosomal information associated with the mapping of the data
`maxChrom`	Highest chromosome to impute.
`smooth`	smoothing parameter for the lowess procedure

Details

There are two main reasons to impute data. One is that given that imputation is reasonable, one can increase the analytical power and improve results. Another, more practical, is that at the moment many widely used fuctions in R do not support missing values. While procedures such as kNN imputations is widely used for gene expression data, it is more powerful to take advantage of the genomic structure of the array CGH data and use a smoother. Note that we perform only one pass os smoothing. If there still remain missing values, they are imputed by the median on the chromosome or chromosomal arm where applicable,

Value

Computes and returns the imputed log2 ratio matrix of the aCGH object.

See Also

`aCGH`, `impute.HMM`.

Examples

```
datadir <- system.file(package = "aCGH")
datadir <- paste(datadir, "/examples", sep="")

clones.info <-
      read.table(file = file.path(datadir, "clones.info.ex.txt"),
                 header = TRUE, sep = "\t", quote="", comment.char="")
log2.ratios <-
      read.table(file = file.path(datadir, "log2.ratios.ex.txt"),
                 header = TRUE, sep = "\t", quote="", comment.char="")
ex.acgh <- create.aCGH(log2.ratios, clones.info)

## Imputing the log2 ratios

log2.ratios.imputed(ex.acgh) <- impute.lowess(ex.acgh)
```

`mergeFunc` *Funtion to merge states based on their state means*

Description

`mergeFunc` takes the output of hmm.run.func (or find.hmm.states) with a particular model selection criterion and iteratively merges the states with means closer than a supplied threshold. `mergeHmmStates` is a frontend for `mergeFunc` using `aCGH` object.

Usage

```
mergeHmmStates(aCGH.obj, model.use = 1, minDiff = 0.25)
mergeFunc(statesres, minDiff = 0.1)
```

Arguments

`aCGH.obj`	Object of class `aCGH`.
`statesres`	the sublist of the states.hmm list output from `find.hmm.states` for a given model selection crterion
`minDiff`	The states whose predicted values are less than minDiff apart are merged into one state and all the predicited values are recomputed.
`model.use`	Model selection criterion to use, See `find.hmm.states`.

Details

This function is intended to reduce effect of the possible small magnitude technological artifacts on the structure determination.

Value

List containing states.hmm object is returned.

Author(s)

Jane Fridlyand

References

Application of Hidden Markov Models to the analysis of the array CGH data, Fridlyand et.al., *JMVA*, 2004

See Also

aCGH, find.hmm.states

---

mergeLevels *mergeLevels*

Description

Merging of predicted levels for array CGH data and similar.

Usage

```
mergeLevels(vecObs,vecPred,pv.thres=0.0001,ansari.sign=0.05,thresMin=0.05,thresMax=0.5,verbose=1,sc
```

Arguments

vecObs	Vector of observed values, i.e. observed log2-ratios
vecPred	Vector of predicted values, i.e. mean or median of levels predicted by segmentation algorithm
pv.thres	Significance threshold for Wilcoxon test for level merging
ansari.sign	Significance threshold for Ansari-Bradley test
thresMin	merge if segment medians are closer than thresMin , defaiult is 0.05
thresMax	don't merge if segment medians are further than thresMax (unless needs to be merged for a different reason: wilcoxon test), default is .5
verbose	if 1, progress is printed
scale	whether thresholds are on the log2ratio scale and thus need to be converted to the copy number. default is TRUE

Details

mergeLevels takes a vector of observed log2-ratios and predicted log2ratios and merges levels that are not significantly distinct.

Value

vecMerged	Vector with merged values. One merged value returned for each predicted/observed value
mnNow	Merged level medians
sq	Vector of thresholds, the function has searched through to find optimum. Note, these thresholds are based on copy number transformed values
ansari	The p-values for the ansari-bradley tests for each threshold in sq

Note

vecObs and vecPred must have same length and observed and predicted value for a given probe should have same position in vecObs and vedPred. The function assumes that log2-ratios are supplied

Author(s)

Hanni Willenbrock (<Hanni@cbs.dtu.dk>) and Jane Fridlyand (<jfridlyand@cc.ucsf.edu>)

References

Willenbrock H, Fridlyand J. (2005). A comparison study: applying segmentation to array CGH data for downstream analyses. Bioinformatics. 2005 Sep 14; [Epub ahead of print]

Examples

```
# Example data of observed and predicted log2-ratios
vecObs <- c(rep(0,40),rep(0.6,15),rep(0,10),rep(-0.4,20),rep(0,15))+rnorm(100,sd=0.2)
vecPred <- c(rep(median(vecObs[1:40]),40),rep(median(vecObs[41:55]),15),
 rep(median(vecObs[56:65]),10),rep(median(vecObs[66:85]),20),rep(median(vecObs[86:100]),15))

# Plot observed values (black) and predicted values (red)
plot(vecObs,pch=20)
points(vecPred,col="red",pch=20)

# Run merge function
merge.obj <- mergeLevels(vecObs,vecPred)

# Add merged values to plot
points(merge.obj$vecMerged,col="blue",pch=20)

# Examine optimum threshold
merge.obj$sq
```

plotFreqStat *frequency plots and significance analysis*

Description

The main application of this function is to plot the frequency of changes.

Usage

```
plotFreqStat(aCGH.obj, resT = NULL, pheno = rep(1, ncol(aCGH.obj)),
             chrominfo = human.chrom.info.Jul03,
             X = TRUE, Y = FALSE,
             rsp.uniq = unique(pheno),
             all = length(rsp.uniq) == 1 && is.null(resT),
             titles = if (all) "All Samples" else rsp.uniq,
             cutplot = 0, thres = .25, factor = 2.5, ylm = c(-1, 1),
             p.thres = c(.01, .05, .1), numaut = 22, onepage = TRUE,
             colored = TRUE)
```

Arguments

aCGH.obj	Object of class aCGH
resT	Data frame having the same structure as the result of applying mt.maxT or mt.minP functions from Bioconductor's multtest package for multiple testing. The result is a data frame including the following 4 components: 'index', 'teststat', 'rawp' and 'adjp'.
pheno	phenotype to compare.
chrominfo	Chromosomal information. Defaults to human.chrom.info.Jul03
X	Include X chromosome? Defaults to yes.
Y	Include Y chromosome? Defaults to no.
rsp.uniq	rsp.uniq specified the codes for the groups of interest. Default is the unique levels of the phenotype. Not used when all is T.
all	all specifies whether samples should be analyzed by subgroups (T) or together (F).
titles	titles names of the groups to be used. Default is the unique levels of the pheno.
cutplot	only clones with at least cutplot frequency of gain and loss are plotted.
thres	thres is either a vector providing unique threshold for each sample or a vector of the same length as number of samples (columns in data) providing sample-specific threshold. If aCGH.obj has non-null sd.samples, then thres is automatically replaced by factor times madGenome of aCGH object. Clone is considered to be gained if it is above the threshold and lost if it below negative threshold. Used for plotting the gain/loss frequency data as well as for clone screening and for significance analysis when threshold is TRUE.Defaults to 0.25

factor	factor specifies the number by which experimental variability should be multiplied. used only when sd.samples(aCGH.obj) is not NULL or when factor is greater than 0. Defaults to 2.5
ylm	ylm vertical limits for the plot
p.thres	p.thres vector of p-value ciut-off to be plotted. computed conservatively as the threshold corresponding to a given adjusted p-value.
numaut	numaut number of the autosomes
onepage	onepage whether all plots are to be plotted on one page or different pages. When more than 2 groups are compared, we recommend multiple pages.
colored	Is plotting in color or not? Default is TRUE.

Examples

```
data(colorectal)

## Use mt.maxT function from multtest package to test
## differences in group means for each clone grouped by sex
colnames(phenotype(colorectal))
sex <- phenotype(colorectal)$sex
sex.na <- !is.na(sex)
colorectal.na <- colorectal[ ,sex.na, keep = TRUE ]
dat <- log2.ratios.imputed(colorectal.na)
resT.sex <- mt.maxT(dat, sex[sex.na], test = "t", B = 1000)

## Plot the result along the genome
plotFreqStat(colorectal.na, resT.sex, sex[sex.na],
             titles = c("Male", "Female"))

## Adjust the p.values from previous exercise with "fdr"
## method and plot them
resT.sex.fdr <- resT.sex
resT.sex.fdr$adjp <- p.adjust(resT.sex.fdr$rawp, "fdr")
plotFreqStat(colorectal.na, resT.sex.fdr, sex[sex.na],
             titles = c("Male", "Female"))

## Derive statistics and p-values for testing the linear association of
## age with the log2 ratios of each clone along the samples

age <- phenotype(colorectal)$age
age.na <- which(!is.na(age))
age <- age[age.na]
colorectal.na <- colorectal[, age.na]
stat.age <- aCGH.test(colorectal.na, age, test = "linear.regression", p.adjust.method = "fdr")

#separate into two groups: < 70 and > 70 and plot freqeuncies of gain and loss
#for each clone. Note that statistic plotted corresponds to linear coefficient
#for age variable

plotFreqStat(colorectal.na, stat.age, ifelse(age < 70, 0, 1), titles =
             c("Young", "Old"), X = FALSE, Y = FALSE)
```

plotGenome *Plots the genome*

Description

Basic plot of the log2 ratios for each array ordered along the genome.

Usage

```
plotGenome(aCGH.obj, samples = 1:num.samples(aCGH.obj), naut = 22,
           Y = TRUE, X = TRUE, data = log2.ratios(aCGH.obj),
           chrominfo = human.chrom.info.Jul03,
           yScale = c(-2, 2), samplenames = sample.names(aCGH.obj),
           ylb = "Log2Ratio")
```

Arguments

aCGH.obj	an object of class aCGH
samples	vector containing indeces of the samples to be plotted.
naut	number of autosomes in the organism
Y	TRUE if chromosome Y is to be plotted, FALSE otherwise
X	TRUE if chromosome X is to be plotted, FALSE otherwise
data	a matrix containing values to use for plotting. defaults to the `log2.ratios(aCGH.obj)`.
chrominfo	a chromosomal information associated with the mapping of the data.
yScale	Minimum y-scale to use for plotting. Scale is expanded if any of the values exceed the positive or negative limit.
samplenames	sample names.
ylb	label for the Y-axis.

See Also

aCGH

Examples

```
#plot samples in the order of descending quality
data(colorectal)
order.quality <- order(sd.samples(colorectal)$madGenome)
pdf("plotGenome.orderByQuality.pdf")
par(mfrow=c(2,1))
for(i in order.quality)
   plotGenome(colorectal, samples = i, Y = FALSE)
dev.off()
```

`plotHmmStates` *Plotting the estimated hmm states and log2 ratios for each sample.*

Description

This function displays the estimated hmm states and log2 ratios for each sample.

Usage

```
plotHmmStates(aCGH.obj, sample.ind, chr = 1:num.chromosomes(aCGH.obj),
             statesres = hmm.merged(aCGH.obj), maxChrom = 23,
             chrominfo = human.chrom.info.Jul03, yScale = c(-2, 2),
             samplenames = sample.names(aCGH.obj))
```

Arguments

`aCGH.obj`	object of class aCGH
`sample.ind`	index of the sample to be plotted relative to the data matrix (i.e. column index in the file)
`statesres`	matrix containing states informations. defaults to the states selected using the first model selection criterionof `aCGH.obj`
`chr`	vector of chromosomes to be plotted
`yScale`	specified scale for Y-exis
`maxChrom`	highest chromosome to show
`chrominfo`	a chromosomal information associated with the mapping of the data
`samplenames`	vector of sample names

Details

Each chromosome is plotted on a separate page and contains two figures. The top figure shows the observed log2ratios and the bottom figure shows predicted values for all clones but outliers which show observed values. The genomic events are indicated on both figures as following. The first clone after transition is indicated with solid blue line and the last clone after transitions is shown with dotted green line. Focal aberrations clones are colored orange, amplifications are colored red and outliers are yellow.

Author(s)

Jane Fridlyand

References

Application of Hidden Markov Models to the analysis of the array CGH data, Fridlyand et.al., *JMVA*, 2004

See Also

aCGH find.hmm.states plotGenome

Examples

```
data(colorectal)
plotHmmStates(colorectal, 1)
```

---

plotSummaryProfile *plotSummaryProfile*

Description

This function display the genomic events and tests for the differences between groups if they are specified.

Usage

```
plotSummaryProfile(aCGH.obj,
                   response = as.factor(rep("All", ncol(aCGH.obj))),
                   titles = unique(response[!is.na(response)]),
                   X = TRUE, Y = FALSE, maxChrom = 23,
                   chrominfo = human.chrom.info.Jul03,
                   num.plots.per.page = length(titles),
                   factor = 2.5, posThres=100, negThres=-0.75)
```

Arguments

aCGH.obj	an object of aCGH class.
response	phenotype to compare. defaults to all the samples being analyzed together.
titles	titles for the groups, defaults to the name of the response
X	logical indicating whether X needs to be shown
Y	logical indicating whether Y needs to be shown
maxChrom	this parameter controls how many chromosomes will be plotted, from 1 to maxChrom
chrominfo	a chromosomal information associated with the mapping of the data
num.plots.per.page	number of frequency plots per page. Default is the number of groups
factor	factor specifies the number by which experimental variability should be multiples. Used only when tumor specific variability in aCGH.obj is not NULL. Defaults to 2.5
posThres	Threshold for gain. Set very high for homozygous deletion
negThres	Threshold for homozygous deletion

Details

This function utilizes output of the `find.genomic.events` by plotting it and testing between groups. The test are performed using kruskal-wallis rank test.

See Also

`aCGH` `find.genomic.events`

Examples

```
data(colorectal)

## Plotting summary of the sample profiles
plotSummaryProfile(colorectal)
```

---

states.hmm.func — *A function to fit unsupervised Hidden Markov model*

Description

This function is a workhorse of `find.hmm.states`. It operates on the individual chromosomes/samples and is not called directly by users.

Usage

```
states.hmm.func(sample, chrom, dat, datainfo = clones.info, vr = 0.01,
                maxiter = 100, aic = FALSE, bic = TRUE, delta = 1,
                nlists = 1, eps = .01, print.info = FALSE,
                diag.prob = .99)
```

Arguments

`sample`	sample identifier
`chrom`	chromosome identifier
`dat`	dataframe with clones in the rows and samples in the columns
`datainfo`	dataframe containing the clones information that is used to map each clone of the array to a position on the genome. Has to contain columns with names Clone/Chrom/kb containing clone names, chromosomal assignment and kb positions respectively
`vr`	Initial experimental variance
`maxiter`	Maximum number of iterations
`aic`	TRUE or FALSE variable indicating whether or nor AIC criterion should be used for model selection (see DETAILS)
`bic`	TRUE or FALSE variable indicating whether or nor BIC criterion should be used for model selection (see DETAILS)

delta	numeric vector of penalty factors to use with BIC criterion. If BIC is true, delta=1 is always calculated (see DETAILS)
nlists	defaults to 1 when aic=TRUE, otherwise > 1
eps	parameter controlling the convergence of the EM algorithm.
print.info	print.info = T allows diagnostic information to be printed on the screen.
diag.prob	parameter controlling the construction of the initial transition probability matrix.

See Also

aCGH

---

summarize.clones *Extracting summary information for all clones*

Description

summarize.clones function is the text equivalent of plotFreqStat function - it summarizes the frequencies of changes for each clone accross tumors and when available assigns statistics. The resulting table can be easily exported.

Usage

```
summarize.clones(aCGH.obj, resT = NULL, pheno = rep(1, ncol(aCGH.obj)), rsp.uniq = unique(pheno), thres
```

Arguments

aCGH.obj	aCGH.obj object here
resT	Data frame having the same structure as the result of applying mt.maxT or mt.minP functions from Bioconductor's multtest package for multiple testing. The result is a data frame including the following 4 components: 'index', 'teststat', 'rawp' and 'adjp'.Default is the unique levels of the phenotype. Not used when all is TRUE.
pheno	phenotype to compare
rsp.uniq	rsp.uniq specified the codes for the groups of interest. Default is the unique levels of the phenotype. Not used when all is TRUE.
thres	thres is either a vector providing unique threshold for each sample or a vector of the same length as number of samples (columns in data) providing sample-specific threshold. If aCGH.obj has non-null sd.samples, then threshold is automatically replaced by tumor-specific sd multiplied by factor. Clone is considered to be gained if it is above the threshold and lost if it below negative threshold. Defaults to 0.25
factor	factor specifies the number by which experimental variability should be multiples. used only when tumor specific variability in aCGH.obj is not NULL. Defaults to 2.5

`all`	`all` specifies whether samples should be analyzed by subgroups (TRUE) or together (FALSE)
`titles`	`titles` names of the groups to be used. Default is the unique levels of the `pheno`.

Value

Returns matrix containg the following information for each clones: annotation (same as in `clones.info`), number and proportion of samples where clone is present,gained and lost; and the same in each group if more than one group. Additionally, if significance comparison has been done, value of the statistic, unadjusted p-value and adjusted p-values are included for each clone.

Author(s)

Jane Fridlyand

See Also

`plotFreqStat`, `aCGH`

Examples

```
data(colorectal)
summarize.clones(colorectal)
```

`threshold.func` *Function to indicate gain or loss for each clone for each sample*

Description

This function outputs a matrix containing gain/loss indicator for each clone and sample.

Usage

```
threshold.func(dat, posThres, negThres = NULL)
```

Arguments

`dat`	log2ratios of the relevant array CGH object
`posThres`	Global or sample-specific threshold for gain
`negThres`	Global or sample-specific threshold for loss. Defaults to -posThres

Value

Returns a matrix with a row for each clone and column for each sample. "1" indicates gain and "-1" indicates loss.

Author(s)

Jane Fridlyand, Ritu Roydasgupta

Examples

```
data(colorectal)

factor <- 3
tbl <- threshold.func(log2.ratios(colorectal),posThres=factor*(sd.samples(colorectal)$madGenome))
rownames(tbl) <- clone.names(colorectal)
colnames(tbl) <- sample.names(colorectal)
tbl[1:5,1:5]
```

Index