Package 'CHAT'

February 19, 2015

Type Package

Title Clonal Heterogeneity Analysis Tool

Version 1.1

Date 2014-02-10

Author Bo Li

Maintainer Bo Li <libo@umich.edu>

Description CHAT is a collection of tools developed for tumor subclonality analysis using high density DNA SNP array data and sequencing data. The pipeline consists of four major compartments: 1) tumor aneuploid genome proportion (AGP) calculation and ploidy estimation. 2) segment-specific AGP calculation and absolute copy number estimation for somatic CNAs. 3) cancer cell fraction correction for somatic SNVs in clonal or subclonal sCNA regions. 4) number of subclones estimation using Dirichlet process prior followed by MCMC approach.

License GPL (>= 2)

Depends R (>= 2.10), parallel, DNAcopy, DPpackage

Repository CRAN

URL http://sourceforge.net/p/clonalhetanalysistool/wiki/CHAT/

NeedsCompilation no

Date/Publication 2014-08-09 00:00:40

R topics documented:

`CHAT-package` *Clonal Heterogeneity Analysis Tool*

Description

CHAT is a collection of methods developed for researchers in the field of cancer genomics to study the intra-heterogeneity of tumors. It is applied on both high-density SNP array data and next generation sequencing data. CHAT estimates tumor purity, ploidy on the genome-wide level. It can also assign each somatic CNA or mutation its fraction of cancer cell carriers, namely, sAGP or CCF.

Details

Package:	CHAT
Type:	Package
Version:	1.0
Date:	2014-02-10
License:	GPL(>=2)

Author(s)

Bo Li <libo@umich.edu>

References

Li et al., Genomic estimates of aneuploid content in Glioblastoma Multiforme and imporved classification. _Clinical Cancer Research_, 2012.

Li et al., A general framework for characterizing tumor subclonality using DNA sequencing and SNP array data. 2014. Submitted to Genome Biology.

`A0SD.BAF` *B-allele frequencies of selected chromosomes for TCGA sample A1-A0SD.*

Description

This is partial level 2 bi-allele copy number data selected from The Cancer Genome Atlas Breast Carcinoma 800K Genome-wide SNP 6.0 array for sample A1-A0SD in the format of B-allele frequency.

Usage

```
data(A0SD.BAF)
```

References

The Cancer Genome Atlas Network, Comprehensive molecular portraits of human breast tumours, _Nature_, 2012

A0SD.LRR *Copy number information of selected chromosomes for TCGA sample A1-A0SD.*

Description

This is partial level 2 bi-allele copy number data selected from The Cancer Genome Atlas Breast Carcinoma 800K Genome-wide SNP 6.0 array for sample A1-A0SD in the format of DNA copy number.

Usage

```
data(A0SD.LRR)
```

References

The Cancer Genome Atlas Network, Comprehensive molecular portraits of human breast tumours, _Nature_, 2012

ApproxBIC *Approximate BIC calculation*

Description

Calculate approximate BIC for a given distribution and number of free parameters in the model.

Usage

```
ApproxBIC(Y, fit.x, fit.y, n)
```

Arguments

Y	input data vector, distribution of which is estimated.
fit.x	x coordinates of the fitted probability density function for Y
fit.y	y coordinates of the fitted probability density function for Y
n	number of parameters in the model

Details

This function is a follow-up of MCMC fitting of a complicated distribution. fit.x and fit.y come from the fitting of Dirichlet process prior on sAGP or CCF distributions.

Value

numeric, BIC score.

Author(s)

Bo Li

Examples

```
data(mcmc)
data(prior)
library(DPpackage)
Y=c(rnorm(40,0.4,0.05),rnorm(40,0.8,0.05))
vv=which(Y>0.05&Y<1)
fit=DPdensity(Y,status=TRUE,mcmc=mcmc,prior=prior)
BIC=ApproxBIC(Y,fit$x1,fit$dens,length(prior))
```

Dist *Distance from point to a line*

Description

Compute point to line distance, internal use.

Usage

```
Dist(x0, y0, A, B, C)
```

Arguments

x0	x coordinate of the point
y0	y coordinate of the point
A,B,C	coefficient for a line Ax+By+C=0.

Value

numeric value of the distance

Author(s)

Bo Li

Examples

```
x0=0.12
y0=0.45
d=Dist(x0,y0,1,1,-0.2)
```

DPfitSamples *MCMC fitting of sAGP values*

Description

This function is a wrapper for `getDPfit()` by analyzing batch samples for sAGP values.

Usage

```
DPfitSamples(dd, alpha = 0.05, low.thr = 0.05, min.peaksize = 10, prior, mcmc, nt=FALSE)
```

Arguments

`dd`	numeric matrix, returned from `getsAGP` or `getSegPurity()`
`alpha`	significant level.
`low.thr`	values below this threshold in sAGP will be omitted.
`min.peaksize`	minimum number of segments each peak must contain.
`prior`	a list of prior parameters required for `DPdensity`. An example is data(prior).
`mcmc`	a list of parameters required to run MCMC for `DPdensity`. An example is data(mcmc).
`nt`	logical. If TRUE, multi-thread processing is performed.

Value

A list containing the following elements:

`DD`	input dd with additional peak information.
`Labels`	a vector assigning each sample to model 0, 1 or 2. See `getDPfit()` for more details.
`Pval`	P value
`par`	parameters fitted for the distribution of sAGP value from each sample.

Author(s)

Bo Li

Examples

```
data(A0SD.BAF)
data(A0SD.LRR)
## DNA segmentation
seg.dat=c()
for(CHR in c(8,9,10)){
baf=A0SD.BAF[A0SD.BAF[,2]==CHR,]
lrr=A0SD.LRR[A0SD.LRR[,2]==CHR,]
```

```
x=getSegChr(baf,lrr)
seg.dat=rbind(seg.dat,x)
}
dd.dat=seg.dat[,2:8]
rownames(dd.dat)=seg.dat[,1]
mode(dd.dat)='numeric'
save(dd.dat,file='A0SDseg.Rdata')
para=getPara()
para$datafile='A0SDseg.Rdata'
para$savefile='A0SD-AGP.txt'
para$is.normalize=FALSE
## AGP estimation
getAGP(para=para)
para.s=getPara.sAGP()
para.s$inputdata='A0SDseg.Rdata'
para.s$purityfile='A0SD-AGP.txt'
para.s$savedata='A0SD-sAGP.Rdata'
## sAGP estimation
getsAGP(para=para.s)
## Perform MCMC fitting
load('A0SD-sAGP.Rdata')
data(mcmc)
data(prior)
temp=DPfitSamples(new.dd,prior=prior,mcmc=mcmc)
```

`findRobustPeaks` *Find well-separated peaks*

Description

This function is a follow-up of `getPeaks()` by merging closely located ambiguous peaks together.

Usage

```
findRobustPeaks(fit, thr = 1.5)
```

Arguments

`fit`	a density fit returned by `density()`
`thr`	threshold to separate adjacent peaks. See details.

Details

If two peaks are closely located, one way to tell if they are actually one peak or otherwise is by comparing the peak height with peak width. By assuming both peaks are normally distributed, this is equivalent to compare the peak height with the height of the local minima (h) between them. `thr` is the minimum ratio of the lower peak height over h. If this ratio is larger than `thr`, the two peaks are considered separated.

Value

a numeric matrix in the same format as returned by `getPeaks()`

Author(s)

Bo Li

Examples

```
data(mcmc)
data(prior)
library(DPpackage)
Y=c(rnorm(40,0.4,0.05),rnorm(40,0.8,0.05))
vv=which(Y>0.05&Y<1)
fit=DPdensity(Y,status=TRUE,mcmc=mcmc,prior=prior)
findRobustPeaks(fit)
```

`getAGP` *AGP estimation main function*

Description

This function is a wrapper for `getSampleAGP()` by taking multiple samples and perform batch estimation.

Usage

```
getAGP(para)
```

Arguments

`para`	list, parameters returned from `getPara()`

Value

A tab-delinimated plain text file with the following fields: sample ID, estimated AGP, (standard deviation), percent genome with CNA, percent on point, percent on regression line, ploidy type, tumor ploidy, overall ploidy, percent of genome amplified, percent of genome deleted, percent of genome with hemizygous deletion, percent of genome with cn-LOH, (95 percent CI lower boundary),(median),(95 percent CI higher boundary).

Author(s)

Bo Li

Examples

```
data(A0SD.BAF)
data(A0SD.LRR)
seg.dat=c()
for(CHR in c(8,9,10)){
baf=A0SD.BAF[A0SD.BAF[,2]==CHR,]
lrr=A0SD.LRR[A0SD.LRR[,2]==CHR,]
x=getSegChr(baf,lrr)
seg.dat=rbind(seg.dat,x)
}
dd.dat=seg.dat[,2:8]
rownames(dd.dat)=seg.dat[,1]
mode(dd.dat)='numeric'
save(dd.dat,file='A0SDseg.Rdata')
para=getPara()
para$datafile='A0SDseg.Rdata'
para$savefile='A0SD-AGP.txt'
para$is.normalize=FALSE
getAGP(para=para)
```

---

getAmpDel *Fraction of amplification or deletion in the genome*

Description

This function returns the fraction of amplified or deleted regions in the genome.

Usage

```
getAmpDel(oo, sam.dat)
```

Arguments

oo	Origin Cluster, as returned from `getOrigin()`
sam.dat	numeric matrix, as returned from `getSeg()` or `getSegChr()`

Value

a vector containing two numeric numbers: fraction of amplification and deletion.

Author(s)

Bo Li

Examples

```
data(A0SD.BAF)
data(A0SD.LRR)
seg.dat=c()
for(CHR in c(8,9,10)){
baf=A0SD.BAF[A0SD.BAF[,2]==CHR,]
lrr=A0SD.LRR[A0SD.LRR[,2]==CHR,]
x=getSegChr(baf,lrr)
seg.dat=rbind(seg.dat,x)
}
dd.dat=seg.dat[,2:8]
rownames(dd.dat)=seg.dat[,1]
mode(dd.dat)='numeric'
para=getPara()
oo=getOrigin(dd.dat,para=para)
getAmpDel(oo,dd.dat)
```

---

getBAFmean *Compute mean of BAF*

Description

Using bi-modal Guassian fitting to calculate folded B-allele-frequency.

Usage

```
getBAFmean(bb)
```

Arguments

bb	numeric vector, original B-allele-frequency data from a given DNA segment.

Value

a numeric number for folded BAF estimate.

Author(s)

Bo Li

Examples

```
## generate BAF values symmetric around 0.5, resembling real data in a CNA region
y1<-rnorm(100,0.3,0.1)
y2<-rnorm(100,0.7,0.1)
y<-c(y1,y2)
## compute folded BAF
```

```
getBAFmean(y)

## generate BAF values distributed around 0.5, resembling real data in a non-CNA region
y<-rnorm(200,0.5,0.2)
## compute folded BAF
getBAFmean(y)
```

`getCanonicalLines` *Regression lines on canonical points with same number of minor alleles*

Description

This function performs linear regression on canonical points with same number of minor alleles. This is an approximation of the real data grid to achieve faster computational speed.

Usage

```
getCanonicalLines(oo, p, type = 1, para)
```

Arguments

`oo`	Origin Cluster, returned by `getOrigin()`
`p`	numeric, AGP value. see `getGrid()` for details.
`type`	integer, ploidy indicator. 1, diploid; 2, tetraploid; 3, hexaploid
`para`	list, parameters returned from `getPara()`

Value

A list including the following elements:

`b`	slope of the line
`k`	intercept of the line
`oo`	Origin Cluster

Author(s)

Bo Li

Examples

```
oo=NULL
oo$x0=0.02
oo$y0=0.1
oo$list=c()
cali=getCanonicalLines(oo,0.8,para=getPara())
```

getCCF *CCF estimation main function*

Description

This is a wrapper for `getSampleCCF()` by allowing batch sample processing.

Usage

```
getCCF(VCFdir, sAGPfile, output = "new.vcf", nt = FALSE,
nc = 10, tc = 11, AD = 3, filter = TRUE,TCGA=TRUE)
```

Arguments

`VCFdir`	directory where the VCF file for all samples is saved.
`sAGPfile`	Rdata file saved by `getsAGP()`
`output`	name of output VCF file.
`nt`	logical, if multi-thread processing is applied. This option is system dependent.
`nc`	the column index in VCF file which coverage information for normal control sample is placed.
`tc`	the column index in VCF file which coverage information for tumor sample is placed.
`AD`	the index of Allele Depth field placed in a ';' deliminated string of sample coverage information.
`filter`	logical. If TRUE, the 'FILTER' column in the VCF file is assumed to have been pre-processed and for variants passed QC, this field is set 'PASS'.
`TCGA`	If you are working with The Cancer Genome Atlas datasets, set this to be TRUE. The sample identifier is assumed to be in format described in https://wiki.nci.nih.gov/display/TCGA/TCGA+Barcode

Details

In the returned VCF, the field INFO is modified to include all the inferences. The additional fields in the INFO columns of the VCF file include (in exactly this order): sample ID, alternative allele count in tumor, total coverage in tumor, alternative allele count in control, total coverage in control, CCF estimation, standard deviation of CCF, sAGP, nb, nt and lineage scenario.

Value

NULL. A new VCF file containing all the somatic mutations from all the VCF files will be generated.

Author(s)

Bo Li

Examples

```
## Not run:
## This is not executable. User must create their own VCF directory.
VCFdir='VCF/'
## Slow. Run with caution.
data(A0SD.BAF)
data(A0SD.LRR)
## DNA segmentation
seg.dat=c()
for(CHR in c(8,9,10)){
baf=A0SD.BAF[A0SD.BAF[,2]==CHR,]
lrr=A0SD.LRR[A0SD.LRR[,2]==CHR,]
x=getSegChr(baf,lrr)
seg.dat=rbind(seg.dat,x)
}
dd.dat=seg.dat[,2:8]
rownames(dd.dat)=seg.dat[,1]
mode(dd.dat)='numeric'
save(dd.dat,file='A0SDseg.Rdata')
para=getPara()
para$datafile='A0SDseg.Rdata'
para$savefile='A0SD-AGP.txt'
para$is.normalize=FALSE
## AGP estimation
getAGP(para=para)
para.s=getPara.sAGP()
para.s$inputdata='A0SDseg.Rdata'
para.s$purityfile='A0SD-AGP.txt'
para.s$savedata='A0SD-sAGP.Rdata'
## sAGP estimation
getsAGP(para=para.s)
## CCF estimation
getCCF(VCFdir,'A0SD-sAGP.Rdata','A0SD-AGP.Rdata')

## End(Not run)
```

---

getCoord *Computes coordinates of a canonical point*

Description

This function calculates the x and y coordinates of a canonical point on BAF-LRR plot.

Usage

```
getCoord(p, x0, y0, nn, nb, nt)
```

Arguments

p	numeric, proportion of aneuploid genome for a DNA segment
x0	numeric, x coordinate for the Origin Cluster
y0	numeric, y coordinate for the Origin Cluster
nn	integer, ploidy indicator. 1, diploid; 2, tetraploid; 3, hexaploid
nb	integer, number of minor alleles
nt	integer, number of total alleles

Value

A list of x and y coordinates.

Author(s)

Bo Li

Examples

```
x0=0.05
y0=0.2
p=0.8
## Calculate the coordinates for copy-neutral-LOH segment
getCoord(p,x0,y0,1,0,2)
```

---

getCosine *Computes cosine value of the angle for two vectors.*

Description

This function implements the Law of Cosines to compute the cosine value of two vectors on BAF-LRR plot.

Usage

```
getCosine(v1, v2)
```

Arguments

v1	numeric vector
v2	numeric vector

Value

numeric cosine value

Author(s)

Bo Li

Examples

```
## See what happens to vectors with 45 degree angle
v1=c(1,1)
v2=c(1,0)
getCosine(v1,v2)
```

---

getDiploidOrigin *Estimate diploid origin in higher ploidy samples*

Description

In higher ploidy samples, the Origin Cluster estimated by getOrigin() usually represents the fraction of genome with balanced gains. This function estimates the position of the unchanged diploid genome on BAF-LRR plot.

Usage

```
getDiploidOrigin(dd, oo, para)
```

Arguments

dd	numeric matrix, as returned from getSegPurity()
oo	list, returned from getOrigin()
para	list, returned from getPara.sAGP()

Value

a new list for diploid genome fraction, in the same format as input oo.

Author(s)

Bo Li

getDistToPath *Find nearest distance from a data point to a canonical line*

Description

For a given point on BAF-LRR plot and a given canonical line, this function finds the nearest point on the line to the data point.

Usage

```
getDistToPath(x0, y0, x, y, type, nb, nt, strictness = 1.1,
thr = 0.001, is.scaling = F, BAF.scale = 8)
```

Arguments

x0	x coordinate for the Origin Cluster
y0	y coordinate for the Origin Cluster
x	x coordinate for the data point
y	y coordinate for the data point
type	integer, ploidy indicator. 1, diploid; 2, tetraploid; 3, hexaploid
nb	number of minor alleles
nt	number of total alleles
strictness	noise tolerance for points with sAGP around 1. See getPara.sAGP() for details.
thr	numeric, see details.
is.scaling	logic. If TRUE, the function will rescale the BAF values so that it has the same dimension as LRR.
BAF.scale	numeric. If is.scaling is TRUE, this is the value to rescale BAF values.

Details

For a given data point A, and a canonical line L, this function first finds a point X on L that is closest to A, and designates the distance from A to X is the closest distance from A to L. This step is achieved by sequentially split the period of L and select the half that is closer to A. The argument thr is the sensitivity threshold to find X during this step.

Value

A list representing a canonical line and containing the following elements:

p.min	sAGP value of the nearest point (X) on L to the data point (A).
dist	distance from A to X.
x.min	x coordinate of X.
y.min	y coordinate of X.

Author(s)

Bo Li

Examples

```
x0=0.05
y0=0.2
x=0.15
y=0.45
## Find X on canonical line for hemizygous amplification:
getDistToPath(x0,y0,x,y,1,1,3)
```

---

getDPfit *MCMC fitting for single sample*

Description

This function implements MCMC with Dirichlet process prior on a numeric vector.

Usage

```
getDPfit(y, alpha = 0.05, low.thr = 0.05,prior,mcmc)
```

Arguments

y	input numeric vector, can be either sAGP or CCF from one sample.
alpha	significance level.
low.thr	values below this threshold in y will be omitted.
prior	a list of prior parameters required for DPdensity. An example is data(prior).
mcmc	a list of parameters required to run MCMC for DPdensity. An example is data(mcmc).

Details

Three models are evaluated in this function. 0) There is not enough events (n<5) to evaluate which model is true. 1) Normal-Uniform mixture and 2) Normal mixture with unknown number of Guassian peaks. The first model is evaluated by SampleNMM(), and the second by MCMC fitting. The two models are compared by BIC scores and a P-value is obtained from likelihood ratio test.

Value

A list is returned. In case of model 0, the list contains:

model	always 0

In case of model 1, the list contains:

PA0	peak information, always equals -1.

A	proportion of Uniform component.
mu	mean of Normal component.
sigma	standard deviation of Normal component.
P	P-value
model	always 1

In case of model 2, the list contains:

PA0	peak information
x,y	density function fitted by MCMC.
P	P value
model	always 2.

Author(s)

Bo Li

Examples

```
data(mcmc)
data(prior)
## model 1
y1=c(runif(50),rnorm(100,0.5,0.1))
getDPfit(y1,prior=prior,mcmc=mcmc)$model
## model 2
y2=c(rnorm(100,0.3,0.05),rnorm(100,0.7,0.05))
getDPfit(y2,prior=prior,mcmc=mcmc)$model
```

---

getGrid *Find all the coordinates for canonical points*

Description

This function computes the grid of canonical points on a BAF-LRR plot

Usage

```
getGrid(x0, y0, p, type = 1, Nc = 10, BAF = "all", para = para)
```

Arguments

x0	x coordinate for the Origin Cluster
y0	y coordinate for the Origin Cluster
p	numeric, an AGP value. See details.
type	integer, ploidy indicator. 1, diploid; 2, tetraploid; 3, hexaploid
Nc	integer, maximum number of total copy number to be considered.
BAF	output option. BAF either takes the value 'all' or an integer number. If BAF=0, only LOH track will be returned; if BAF=1, only canonical points with configuration (1,nt) will be returned; etc.
para	list, parameters returned from `getPara()`

Details

On a BAF-LRR plot, DNA segments with different combinations of minor allele (nb) and total alleles (nt) are located on different positions, namely canonical positions. The coordinates of these positions are affected by both nb, nt and the possible contraction rate towards the origin, which represents the fraction of cells without any copy number alterations in the tumor sample. the input, p is the contraction rate, later will be called aneuploid genome proportion, or AGP.

Value

A numeric matrix with two columns: x and y coordinates of canonical positions.

Author(s)

Bo Li

Examples

```
x0=0.02
y0=0.1
p=0.8
gg=getGrid(x0,y0,p,para=getPara())
## place canonical points on a BAF-LRR plot, with AGP=0.8
plot(0,0,cex=0,xlim=c(0,0.5),ylim=c(-1,3),xlab='BAF',ylab='LRR')
points(gg,pch='*',cex=2,lwd=2)
```

---

getHets *Obtain germline heterozygous markers*

Description

Remove uninformative germline homozygous markers for a DNA segment.

Usage

```
getHets(baf, paired.baf = NULL, thr = 0.18)
```

Arguments

baf	numeric vector, original B-allele frequency from a cancer DNA segment
paired.baf	numeric vector, original B-allele frequency from a paired normal DNA segment, with exactly the same length as baf. See details.
thr	numeric, threshold to call germline heterozygous markers.

Details

When paired.baf is given, any marker has germline BAF>=1-thr or BAF<=thr are considered as homozygous. If paired.baf is missing, then in non-LOH regions, the method uses the same criteria to select heterozygous markers; in LOH regions, 1/3 of the markers are randomly selected as germline heterozygous.

Value

a vector of indices indicating the positions of heterozygous markers in input baf.

Author(s)

Bo Li

Examples

```
library(CHAT)
data(A0SD.BAF)
baf=A0SD.BAF[1:1000,5]
paired.baf=A0SD.BAF[1:1000,6]
vv=getHets(baf,paired.baf)
```

---

getKmeans *K-means clustering of data points*

Description

Performs canonical K-means clustering of data points on BAF-LRR plot. (Internal use)

Usage

```
getKmeans(sam.dat, iter.max = 10, para)
```

Arguments

sam.dat	numeric matrix, segmentation file returned by getSegChr() or getSeg()
iter.max	integer, maximum number of iteration to be performed before convergence
para	a list of parameters returned by getPara()

Value

a kmeans object for internal use.

Author(s)

Bo Li

Examples

```
para=getPara()
## Not run:
data(A0SD.BAF)
data(A0SD.LRR)
chr8.baf=A0SD.BAF[A0SD.BAF[,2]==8,]
chr8.lrr=A0SD.LRR[A0SD.LRR[,2]==8,]
x=getSegChr(chr8.baf,chr8.lrr)
getKmeans(x,para=para)

## End(Not run)
```

---

getLOH *Fraction of loss-of-heterozygosity*

Description

This function returns the fraction of loss-of-heterozygosity regions in the genome

Usage

```
getLOH(cali, sam.dat, para)
```

Arguments

cali	list, returned by getSumDist()
sam.dat	numeric matrix, as returned from getSeg() or getSegChr()
para	list, parameters returned from getPara()

Value

list, inherited from getSumDist() with additional elements:

del.loh	fraction of LOH with deletions
cn.loh	fraction of copy neutral LOH
amp.3	fraction of hemizygous amplification

Author(s)

Bo Li

Examples

```
data(A0SD.BAF)
data(A0SD.LRR)
seg.dat=c()
for(CHR in c(8,9,10)){
baf=A0SD.BAF[A0SD.BAF[,2]==CHR,]
lrr=A0SD.LRR[A0SD.LRR[,2]==CHR,]
x=getSegChr(baf,lrr)
seg.dat=rbind(seg.dat,x)
}
dd.dat=seg.dat[,2:8]
rownames(dd.dat)=seg.dat[,1]
mode(dd.dat)='numeric'
para=getPara()
oo=getOrigin(dd.dat,para=para)
cali=getSampleAGP(dd.dat,oo,para=para)
getLOH(cali,dd.dat,para=para)
```

---

getOrigin *Inference of Origin Cluster*

Description

This function performs K-means on BAF-LRR plot and assign a subset of data points to the Origin Cluster using the concept of consensus clustering.

Usage

```
getOrigin(sam.dat, ntry = 10, concensus.thr = 0.6, para)
```

Arguments

sam.dat	numeric matrix, as returned from getSegChr() or getSeg(), with one sample included.
ntry	integer, number of attempts to perform K-means.
concensus.thr	numeric, rate of concensus required to be included in to Origin Cluster. A data point needs to be assigned to the Origin Cluster >= ntry*concensus.thr times to be included.
para	a list of parameters returned from getPara().

Details

The Origin Cluster represents the unchanged genome in a tumor sample, and it is important for downstream inference. This function implement a two-step approach to accurately assign the membership to data points. First, it performs concensus clustering on the data points and find a minimum set of the Origin Cluster. It then expands the set by iteratively including more data points into it as long as the new point is close to the centroid of the Cluster.

Value

a list containing the following elements:

x0	the x coordinate of the Origin Cluster centroid
y0	the y coordinate of the Origin Cluster centroid
list	indices of the segments belong to Origin Cluster in sam.dat.

Author(s)

Bo Li

Examples

```
para=getPara()

data(A0SD.BAF)
data(A0SD.LRR)
chr8.baf=A0SD.BAF[A0SD.BAF[,2]==8,]
chr8.lrr=A0SD.LRR[A0SD.LRR[,2]==8,]
x=getSegChr(chr8.baf,chr8.lrr)
dd.dat=x[,2:8]
rownames(dd.dat)=x[,1]
mode(dd.dat)='numeric'
oo=getOrigin(dd.dat,para=para)

## See what it looks like on a BAF-LRR plot
plot(dd.dat[,6],dd.dat[,4],pch=19,cex=0.3,xlab='BAF',ylab='LRR',xlim=c(0,0.5),ylim=c(-1,1))
points(x[oo$list,6],dd.dat[oo$list,4],pch=19,cex=0.3,col='yellow')
```

---

getPara *Initialize parameters for AGP estimation*

Description

This function sets default values to parameters required to run AGP estimation.

Usage

```
getPara()
```

Value

The default parameters obtained from this function is essential to run `getAGP()`. These parameters include:

datafile	full path to input data file generated by `getSeg()` or in the same format
savefile	name of the output plain text file to be saved

pngdir	directory to save BAF-LRR plots, ignored if is.png is FALSE
BAFfilter	DNA segments with BAF markers below this value are removed to reduce noise. Default 10
thr.kmeans	convergence threshold for getKmeans(). Default 0.1
thr.originsize	minimum number of markers included in the origin cluster, used by getOrigin(). Default 500
thr.CL	threshold distance to call a data point to be close to a canonical line, used by getSumDist(). Default 0.05
thr.CP	threshold distance to call a data point to be close to a canonical point, used by getSumDist(). Default 0.05
thr.penalty	penalty (in units of number of markers) to increase ploidy. Default 500
std.BAF	standard deviation of BAF markers. Default 0.02
std.LRR	standard deviation of LRR markers. Default 0.08
exclude.chr	chromosomes to be excluded from the analysis. Default NULL
res.r	resampling ratio for bootstrap. Default 0.8
num.tracks	number of canonical tracks to be considered in likelihood estimation. Default 4
is.normalize	if data needs to be normalized first. Default TRUE
is.perm	if bootstrap based null distribution of AGP is estimated, time consuming. Default FALSE
is.png	if BAF-LRR plots are saved. Default TRUE
is.plot	if BAF-LRR plots are shown during the process of AGP estimation. Default TRUE
LRR_correction_del	factor to correct negative LRR values. Default 0.572
LRR_correction_amp	factor to correct positive LRR values. Default 0.553

Author(s)

Bo Li

Examples

```
para=getPara()
## Change threshold for ploidy increase
para$thr.penalty=200
## Skip png plotting
para$is.png=FALSE
```

`getPara.sAGP` *Initialize parameters for sAGP estimation.*

Description

This function sets the default values of parameters required to run sAGP estimation.

Usage

```
getPara.sAGP()
```

Value

The default parameters obtained from this function is essential to run `getsAGP()`. These parameters include:

`inputdata`	full path to input data file generated by `getSeg()` or in the same format
`savedata`	name of the output plain text file to be saved
`pngdir`	directory to save BAF-LRR plots, ignored if is.png is FALSE
`purityfile`	AGP estimation file returned by `getAGP()`
`BAFfilter`	DNA segments with BAF markers below this value are removed to reduce noise. Default 10
`thr.kmeans`	convergence threshold for `getKmeans()`. Default 0.1
`thr.originsize`	minimum number of markers included in the origin cluster, used by `getOrigin()`. Default 500
`thr.penalty`	penalty (in units of number of markers) to increase ploidy. Default 500
`std.BAF`	standard deviation of BAF markers. Default 0.025
`std.LRR`	standard deviation of LRR markers. Default 0.1
`is.normalize`	if data needs to be normalized first. Default TRUE
`LRR_correction_del`	factor to correct negative LRR values. Default 0.572
`LRR_correction_amp`	factor to correct positive LRR values. Default 0.553
`K`	constant parameter for objective function F (see details in `getSegPurity()`). Default 12
`is.plot`	if TRUE, the process of estimating sAGP value for each data point will be shown on a prompted BAF-LRR plot. Default FALSE
`max.cn`	maximum copy number examined. Default ploidy+4
`strictness`	maximum sAGP value considered during optimization. By allowing it to be larger than 1, small noise around 1 could be tolerated. Default 1.1
`is.multicore`	if to use multiple thread to increase computational speed. Default FALSE.

Author(s)

Bo Li

Examples

```
para.s=getPara.sAGP()
## Increase strictness
para.s$strictness=1
## Not run:
## Use multi-thread computing
install.packages('multicore')
library(multicore)
para.s$is.multicore=TRUE

## End(Not run)
```

---

getPeaks *Find peaks on a curve*

Description

Find all the local maxima and their corresponding positions in a given curve.

Usage

```
getPeaks(x, y, sep = 8, thr = 0.01)
```

Arguments

x	x coordinates for the curve.
y	y coordinates for the curve.
sep	number of initial split of the curve.
thr	sensitivity threshold to call a local maxima.

Value

numeric matrix with two columns: position of peaks and peak values.

Author(s)

Bo Li

Examples

```
Y=c(rnorm(50,0.4,0.05),rnorm(50,0.8,0.05))
fit=density(Y)
getPeaks(fit$x,fit$y)
```

`getPermutation` *Null distribution for estimated AGP.*

Description

This function performs weighted bootstrap on segmentation data to obtain null distribution of estimated AGP.

Usage

```
getPermutation(sam.dat, oo, Np = 100, type = 1, para)
```

Arguments

`sam.dat`	numeric matrix, as returned from `getSeg()` or `getSegChr()`
`oo`	Origin Cluster, as returned from `getOrigin()`
`Np`	integer, number of bootstraps.
`type`	integer, ploidy indicator. 1, diploid; 2, tetraploid; 3, hexaploid
`para`	list, parameters returned from `getPara()`

Value

A numeric vector containing all the bootstrap AGP values from Np times of resampling.

Author(s)

Bo Li

Examples

```
## Slow. Run with caution.

data(A0SD.BAF)
data(A0SD.LRR)
seg.dat=c()
for(CHR in c(8,9,10)){
baf=A0SD.BAF[A0SD.BAF[,2]==CHR,]
lrr=A0SD.LRR[A0SD.LRR[,2]==CHR,]
x=getSegChr(baf,lrr)
seg.dat=rbind(seg.dat,x)
}
dd.dat=seg.dat[,2:8]
rownames(dd.dat)=seg.dat[,1]
mode(dd.dat)='numeric'
para=getPara()
oo=getOrigin(dd.dat,para=para)
p.null=getPermutation(dd.dat,oo,para=para)
```

getPloidy *Genomewide average ploidy*

Description

This function returns the genomewide average ploidy of aneuploid fraction and overall of the sample.

Usage

```
getPloidy(p, type = 1, sam.dat, oo)
```

Arguments

p	AGP value. see `getGrid()` for details.
type	integer, ploidy indicator. 1, diploid; 2, tetraploid; 3, hexaploid
sam.dat	numeric matrix, as returned from `getSeg()` or `getSegChr()`
oo	Origin Cluster, as returned from `getOrigin()`

Value

Ploidy of the overall sample and ploidy of the aneuploid proportion are returned.

Author(s)

Bo Li

Examples

```
data(A0SD.BAF)
data(A0SD.LRR)
seg.dat=c()
for(CHR in c(8,9,10)){
baf=A0SD.BAF[A0SD.BAF[,2]==CHR,]
lrr=A0SD.LRR[A0SD.LRR[,2]==CHR,]
x=getSegChr(baf,lrr)
seg.dat=rbind(seg.dat,x)
}
dd.dat=seg.dat[,2:8]
rownames(dd.dat)=seg.dat[,1]
mode(dd.dat)='numeric'
para=getPara()
oo=getOrigin(dd.dat,para=para)
getPloidy(0.5,sam.dat=dd.dat,oo=oo)
```

getsAGP *Segment-specific AGP estimation main function*

Description

This function is a wrapper of `getSegPurity()` and performs sAGP estimation by batch of samples.

Usage

```
getsAGP(para)
```

Arguments

`para`	list, parameters returned from `getPara.sAGP()`

Value

A numeric matrix with name 'new.dd' inherited from `getSeg()` or `getSegChr()` with additional columns: sAGP, nb and nt is saved in the current working directory.

Author(s)

Bo Li

Examples

```
data(A0SD.BAF)
data(A0SD.LRR)
## DNA segmentation
seg.dat=c()
for(CHR in c(8,9,10)){
baf=A0SD.BAF[A0SD.BAF[,2]==CHR,]
lrr=A0SD.LRR[A0SD.LRR[,2]==CHR,]
x=getSegChr(baf,lrr)
seg.dat=rbind(seg.dat,x)
}
dd.dat=seg.dat[,2:8]
rownames(dd.dat)=seg.dat[,1]
mode(dd.dat)='numeric'
save(dd.dat,file='A0SDseg.Rdata')
para=getPara()
para$datafile='A0SDseg.Rdata'
para$savefile='A0SD-AGP.txt'
para$is.normalize=FALSE
## AGP estimation
getAGP(para=para)
para.s=getPara.sAGP()
para.s$inputdata='A0SDseg.Rdata'
```

```
para.s$purityfile='A0SD-AGP.txt'
para.s$savedata='A0SD-sAGP.Rdata'
## sAGP estimation
getsAGP(para=para.s)
```

`getSampleAGP` *AGP inference by sample*

Description

This function performs AGP inference based on canonical positions on BAF-LRR plot for one sample.

Usage

```
getSampleAGP(sam.dat, oo, para)
```

Arguments

`sam.dat`	numeric matrix, as returned from `getSeg()` or `getSegChr()`
`oo`	Origin Cluster, as returned from `getOrigin()`
`para`	list, parameters returned from `getPara()`

Details

AGP values and possible ploidy types of the input sample are screened for best combination which minimizes the total summation of distances from observed data points to the grid of theoretical canonical positions.

Value

A list inherited from the output of `getSumDist()` with additional elements:

`percent.on.track`	For quality control, percent of genome close to at least one regression line.
`percent.on.point`	For quality control, percent of genome close to at least one canonical point.
`percent.change`	For quanlity control, percent of genome with CNA.
`sam.p`	estimated AGP of the sample
`type`	estimated ploidy type of the sample

If `is.perm` is set TRUE in `para`, additional elements are included:

`conf.int`	95 percent confident interval of estimated AGP
`std`	standard deviation of estimated AGP

Author(s)

Bo Li

Examples

```
data(A0SD.BAF)
data(A0SD.LRR)
seg.dat=c()
for(CHR in c(8,9,10)){
baf=A0SD.BAF[A0SD.BAF[,2]==CHR,]
lrr=A0SD.LRR[A0SD.LRR[,2]==CHR,]
x=getSegChr(baf,lrr)
seg.dat=rbind(seg.dat,x)
}
dd.dat=seg.dat[,2:8]
rownames(dd.dat)=seg.dat[,1]
mode(dd.dat)='numeric'
para=getPara()
oo=getOrigin(dd.dat,para=para)
getSampleAGP(dd.dat,oo,para=para)
```

---

getSampleCCF *CCF estimation by sample.*

Description

Finds the Cancer Cell Fraction values of each somatic mutations in one sample.

Usage

```
getSampleCCF(id, dd, VCFdir, thr.coverage = 10, upper.cov=0.99,
nc = 10, tc = 11, AD = 3, filter = TRUE, TCGA=TRUE)
```

Arguments

id	sample identifier. Must be appearing in the VCF file name.
dd	numeric matrix, as returned from `getSegPurity()` or `getsAGP()`
VCFdir	directory where the VCF file for the input sample is saved.
thr.coverage	variants with coverage below this threshold will be removed.
upper.cov	sites with coverage above this percentile will be removed.
nc	the column index in VCF file which coverage information for normal control sample is placed.
tc	the column index in VCF file which coverage information for tumor sample is placed.

AD	the index of Allele Depth field placed in a ';' deliminated string of sample coverage information.
filter	logical. If TRUE, the 'FILTER' column in the VCF file is pre-processed and for variants passed QC, this field is set 'PASS'.
TCGA	If you are working with The Cancer Genome Atlas datasets, set this to be TRUE. The sample identifier is assumed to be in format described in https://wiki.nci.nih.gov/display/TCGA/TCGA+Barcode

Value

a matrix in the format of VCF file for all somatic mutations with updated 'INFO' field. The additional information include: sample ID, coverage of reference allele in tumor, coverage of alternative allele in tumor, coverage of reference allele in control, coverage of alternative allele in control, CCF estimation, standard deviation of CCF, sAGP, nb, nt and lineage scenario.

Author(s)

Bo Li

Examples

```
## Not run:
## This is not executable. User must create their own VCF directory.
id='TCGA.A1.A0SD'
VCFdir='VCF/'
## Slow. Run with caution.
data(A0SD.BAF)
data(A0SD.LRR)
## DNA segmentation
seg.dat=c()
for(CHR in c(8,9,10)){
baf=A0SD.BAF[A0SD.BAF[,2]==CHR,]
lrr=A0SD.LRR[A0SD.LRR[,2]==CHR,]
x=getSegChr(baf,lrr)
seg.dat=rbind(seg.dat,x)
}
dd.dat=seg.dat[,2:8]
rownames(dd.dat)=seg.dat[,1]
mode(dd.dat)='numeric'
save(dd.dat,file='A0SDseg.Rdata')
para=getPara()
para$datafile='A0SDseg.Rdata'
para$savefile='A0SD-AGP.txt'
para$is.normalize=FALSE
## AGP estimation
getAGP(para=para)
para.s=getPara.sAGP()
para.s$inputdata='A0SDseg.Rdata'
para.s$purityfile='A0SD-AGP.txt'
para.s$savedata='A0SD-sAGP.Rdata'
## sAGP estimation
```

```
getsAGP(para=para.s)
load('A0SD-sAGP.Rdata')
getSampleCCF(id,new.dd,VCFdir)

## End(Not run)
```

---

getSeg *Perform DNA segmentation and calculate folded BAF and logR ratios.*

Description

The function is a wrapper for `getSegChr()`, which calculates the folded BAF and logR ratios for each chromosome.

Usage

```
getSeg(BAFfilePath, logRfilePath, sam.col=5, control=TRUE,
thr.hets=0.1, bin = 500, savefile, data.type='copy',cbs=FALSE,
controlOne=0,nt=FALSE)
```

Arguments

`BAFfilePath`	character string, full path to directory where all BAF files are located.
`logRfilePath`	character string, full path to directory where all LRR files are located.
`sam.col`	the index of the column in BAF or LRR files where the first sample starts.
`control`	If TRUE, each tumor sample is paired with normal immediately after it. The columns of the data file is organized : sample_1, control_1, sample_2, control_2 .... If FALSE, each sample serves the control of itself.
`thr.hets`	lower threshold of calling homozygous markers. BAF<=thr.hets or BAF>=1-thr.hets are considered homozygous.
`bin`	integer, number of markers in each bin.
`savefile`	character string, Rdata file to be saved in working directory.
`data.type`	character string chosen from c('copy', 'log'). If 'copy', the value for LRR markers represent the copy number of SNPs. If 'log', the value is log2 based copy number intensity.
`cbs`	if TRUE, circular binary segmentation will be performed using R package 'DNAcopy', instead of binned segmentation.
`controlOne`	default 0. If assigned, must be an integer number indicating the index of one control sample in the sample list. This control will be used for all the samples.
`nt`	logical. If true, multi-thread processing is applied. This option is system dependent.

Details

The input format for BAF and LRR files are the same and as follows: 1) in the directory, there are n tab-deliminated plain text files, where n is the number of chromosomes, with file name chrN.txt, N=1,2,...22. sex and mitochondria chromosomes are currently not included in the analysis. No other file should be included. 2) for tumor/normal paired design (`control=TRUE`), each plain text file must have the following fields: Marker-ID, chromosome, start position, end position, Tumor 1, Paired-normal 1, Tumor 2, Paired-normal 2, ..., etc. For study with tumors matched with one control sample, the order of samples is not required. It is not necessary to sort the genomic positions beforehand.

Value

NULL. An Rdata file with variable name 'dd.dat' will be saved in the working directory with following fields: sample-ID, chromosome, start position, end position, LRR value, number of LRR markers, BAF value, number of BAF markers. The saved file can be immediately used in downstream analysis.

Author(s)

Bo Li

See Also

getSegChr

Examples

```
## Not run:
getSeg(directory_to_BAF_files, directory_to_LRR_files, bin=1000)

## End(Not run)
```

---

getSegChr *Perform binned DNA segmentation by chromosome*

Description

This function computes folded BAF and logR ratios for binned DNA segments for a given chromosome.

Usage

```
getSegChr(bb.chr = NULL, ll.chr = NULL, sam.col=5, control=TRUE,
thr.hets=0.1, bin = 1000, data.type='copy',controlOne=0)
```

Arguments

bb.chr	numeric matrix, original B-allele frequency for a given chromosome for all samples.
ll.chr	numeric matrix, original logR ratio for a given chromosome for all samples.
sam.col	the index of the column in BAF or LRR files where the first sample starts
control	If TRUE, each tumor sample is paired with normal immediately after it. The columns of the data file is organized : sample_1, control_1, sample_2, control_2 .... If FALSE, each sample serves the control of itself.
thr.hets	lower threshold of calling homozygous markers. BAF<=thr.hets or BAF>=1-thr.hets are considered homozygous.
bin	integer, number of markers contained in each bin.
data.type	character string chosen from c('copy', 'log'). If 'copy', the value for LRR markers represent the copy number of SNPs. If 'log', the value is log2 based copy number intensity.
controlOne	default 0. If positive, must be an integer number indicating the index of one control sample in the sample list. This control will be used for all the samples.

Value

a matrix containing the following columns: chromosome, start position, end position, median LRR value, number of LRR markers, folded BAF value, number of germline heterozygous BAF markers, with row names being sample ID.

Author(s)

Bo Li

See Also

getSeg

Examples

```
library(CHAT)
data(A0SD.BAF)
data(A0SD.LRR)
CHR=8
chr8.baf=A0SD.BAF[which(A0SD.BAF[,2]==CHR),]
chr8.lrr=A0SD.LRR[which(A0SD.LRR[,2]==CHR),]
dd=getSegChr(chr8.baf,chr8.lrr)
print(dd[1:4,])
```

getSegChr.CBS *Perform circular binary segmentation by chromosome*

Description

This function performs CBS on B-allele freqeuncies and logR ratios on one chromosome for all the samples and returns merged break points generated by the two datasets.

Usage

```
getSegChr.CBS(bb.chr = NULL, ll.chr = NULL, sam.col=5,
control=TRUE, thr.hets=0.1, data.type = 'copy',
controlOne=0,nt=FALSE)
```

Arguments

bb.chr	numeric matrix, original B-allele frequency for a given chromosome for all samples.
ll.chr	numeric matrix, original logR ratio for a given chromosome for all samples.
sam.col	the index of the column in BAF or LRR files where the first sample starts
control	If TRUE, each tumor sample is paired with normal immediately after it. The columns of the data file is organized : sample_1, control_1, sample_2, control_2 .... If FALSE, each sample serves the control of itself.
thr.hets	lower threshold of calling homozygous markers. BAF<=thr.hets or BAF>=1-thr.hets are considered homozygous.
data.type	character string chosen from c('copy', 'log'). If 'copy', the value for LRR markers represent the copy number of SNPs. If 'log', the value is log2 based copy number intensity.
controlOne	default NA. If assigned, must be an integer number indicating the index of one control sample in the sample list. This control will be used for all the samples.
nt	logic, if TRUE, multi-thread processing will be used. Require R package multicore. Not supported on Windows system.

Value

a matrix containing the following columns: chromosome, start position, end position, median LRR value, number of LRR markers, folded BAF value, number of germline heterozygous BAF markers, with row names being sample ID.

Author(s)

Bo Li

See Also

getSeg

Examples

```
data(A0SD.BAF)
data(A0SD.LRR)
CHR=8
chr8.baf=A0SD.BAF[which(A0SD.BAF[,2]==CHR),]
chr8.lrr=A0SD.LRR[which(A0SD.LRR[,2]==CHR),]
dd=getSegChr.CBS(chr8.baf,chr8.lrr)
print(dd[1:4,])
```

`getSegPurity` *Segment-specific AGP inference by sample*

Description

This function implements the sAGP inference algorithm, by placing each of the data point onto a BAF-LRR plot. AGP inference must be done in prior.

Usage

```
getSegPurity(seg.dat, oo, AGP = 1, type = 1, para, rm.thr = 50, ref.dd = NULL)
```

Arguments

`seg.dat`	numeric matrix, segmentation data returned from getSegChr() or getSeg()
`oo`	list, origin information returned from getOrigin() or getDiploidOrigin()
`AGP`	numeric, AGP value for the sample being evaluated
`type`	integer, ploidy type of tumors. 1, diploid; 2, tetraploid; 3, hexaploid.
`para`	list, parameters returned fro getPara.sAGP()
`rm.thr`	integer, segments with number of BAF markers below this threshold will be removed.
`ref.dd`	numeric matrix, in the same format as sam.dd. If given, no copy number estimation will be performed, and only sAGP will be inferred.

Details

For a data point A on BAF-LRR plot, the algorithm search through all the canonical lines and find the nearest ones to A. Each canonical line corresponds to the contraction path from a canonical point with (nb, nt) towards the origin, where nb is the number of minor allele and nt the number of total alleles. It then uses an empirical objective function F=nt-2*type+K*|sAGP-AGP|, to determine which line to choose. sAGP is the estimated aneuploid fraction of the specific segment for a given canonical line and K the constant parameter set in para. The purpose of this function is to find the most parsimonious combination of nb, nt and sAGP, with parsimony meaning close to genome-wide average.

Value

a numeric matrix with following columns: chromosome, start position, end position, LRR value, number of LRR markers, BAF value, number of BAF markers, sAGP, number of minor alleles, number of total alleles.

Author(s)

Bo Li

Examples

```
data(A0SD.BAF)
data(A0SD.LRR)
seg.dat=c()
for(CHR in c(8,9,10)){
baf=A0SD.BAF[A0SD.BAF[,2]==CHR,]
lrr=A0SD.LRR[A0SD.LRR[,2]==CHR,]
x=getSegChr(baf,lrr)
seg.dat=rbind(seg.dat,x)
}
dd.dat=seg.dat[,2:8]
rownames(dd.dat)=seg.dat[,1]
mode(dd.dat)='numeric'
para.s=getPara.sAGP()
para=getPara()
oo=getOrigin(dd.dat,para=para)
sAGP.dat=getSegPurity(dd.dat,oo,AGP=0.5,para=para.s)
```

---

getSumDist *Compute the summation of distances*

Description

Canonical positions are set on a BAF-LRR plot with given AGP value. This function finds the distance from each data point to its nearest canonical position and returns the summation.

Usage

```
getSumDist(oo, p, sam.dat, method = "both", type = 1, para)
```

Arguments

oo	Origin Cluster, as returned from `getOrigin()`
p	AGP value. see `getGrid()` for details.
sam.dat	numeric matrix, as returned from `getSeg()` or `getSegChr()`

method	string, chosen from 'both','logR' and 'BAF', the column(s) from sam.dat to weigh the distance from a data point to canonical positions. If 'logR' or 'BAF', number of LRR or BAF markers will be used as weight. If 'both', geometric average of both numbers will be used.
type	integer, ploidy indicator. 1, diploid; 2, tetraploid; 3, hexaploid
para	list, parameters returned from `getPara()`

Value

A list inherited from the list returned from `getCanonicalLines()`, with extra elements being:

SD	sum of distances
clist	indices of data points close to at least one regression line.
plist	indices of data points close to at least one canonical position.

Author(s)

Bo Li

Examples

```
data(A0SD.BAF)
data(A0SD.LRR)
seg.dat=c()
for(CHR in c(8,9,10)){
baf=A0SD.BAF[A0SD.BAF[,2]==CHR,]
lrr=A0SD.LRR[A0SD.LRR[,2]==CHR,]
x=getSegChr(baf,lrr)
seg.dat=rbind(seg.dat,x)
}
dd.dat=seg.dat[,2:8]
rownames(dd.dat)=seg.dat[,1]
mode(dd.dat)='numeric'
para=getPara()
oo=getOrigin(dd.dat,para=para)
cali=getSumDist(oo,0.5,dd.dat,para=para)
```

---

getUnifiedMap *Merge adjacent break points*

Description

Two sets of break points from one chromosome, typically from B-allele frequency and logR ratios respectively are inputs, and this function uses input from LRR as template to adjust break point positions in BAF if they are close enough to one of the LRR break points.

Usage

```
getUnifiedMap(seg.L.chr, seg.B.chr, map.chr, thr = 5)
```

Arguments

seg.L.chr	break point positions from LRR segmentation.
seg.B.chr	break point positions from BAF segmentation.
map.chr	SNP positions or starting position of aCGH markers.
thr	if two break points are located within thr number of markers, they will be merged.

Details

Each set of break points contains even number, increasing integers, alternating between start and end positions of a DNA segment.

Value

a vector containing the merged break point positions.

Author(s)

Bo Li

Examples

```
seg.L<-c(1,10,11,100,101,120)
seg.B<-c(1,11,12,60,61,120)
map.chr<-1:130
getUnifiedMap(seg.L,seg.B,map.chr)
```

---

is.nearCP *Near canonical position*

Description

Given coordinates of a data point and canonical position grid, this function tells if the data point is close to at least one canonical positions on the BAF-LRR plot.

Usage

```
is.nearCP(seg.x, seg.y, gg, para)
```

Arguments

seg.x	x coordinate of the data point.
seg.y	y coordinate of the data point.
gg	canoincal position grid as returned from getGrid()
para	list, parameters returned from getPara()

Value

logical

Author(s)

Bo Li

Examples

```
x0=0.02
y0=0.1
p=0.8
gg=getGrid(x0,y0,p,para=getPara())
is.nearCP(0.42,0.1,gg,para=getPara())
```

---

mcmc *MCMC parameters.*

Description

A pre-selected set of parameters required to run DPdensity()

Usage

```
data(mcmc)
```

Format

The format is: List of 4 $ nburn : num 10000 $ nskip : num 4 $ ndisplay: num 100 $ nsave : num 1000

MergeBreakPointsByChr *Merge break points from two sets*

Description

Two sets of break points from one chromosome, typically from B-allele frequency and logR ratios respectively are inputs, and this function merges the two sets as one.

Usage

```
MergeBreakPointsByChr(chr, id, seg.B, seg.L, map.chr, thr = 5)
```

Arguments

chr	chromosome number
id	sample identifier
seg.B	segmentation matrix for BAF as returned by segment() from 'DNAcopy'
seg.L	segmentation matrix for LRR as returned by segment() from 'DNAcopy'
map.chr	SNP positions or starting position of aCGH markers.
thr	if two break points are located within thr number of markers, they will be merged.

Value

a new matrix with merged segmentations.

Author(s)

Bo Li

Examples

```
seg.L<-cbind('temp',1,matrix(c(1,10,11,100,101,120),ncol=2,byrow=TRUE))
seg.B<-cbind('temp',1,matrix(c(1,11,12,60,61,120),ncol=2,byrow=TRUE))
map.chr<-1:130
MergeBreakPointsByChr(1,'temp',seg.B,seg.L,map.chr)
```

NormalizeLRR *Normalize LRR values*

Description

This function performs correction on both negative and positive LRR values, based on factors given in getPara().

Usage

```
NormalizeLRR(x, para)
```

Arguments

x	numeric matrix, typically one sample from the complete data matrix returned from getSeg().
para	list, parameters returned by getPara().

Value

A new numeric data matrix with LRR normalized.

Author(s)

Bo Li

Examples

```
para=getPara()
## Not run:
data(A0SD.BAF)
data(A0SD.LRR)
chr8.baf=A0SD.BAF[A0SD.BAF[,2]==8,]
chr8.lrr=A0SD.LRR[A0SD.LRR[,2]==8,]
x=getSegChr(chr8.baf,chr8.lrr)
NormalizeLRR(x,para=para)

## End(Not run)
```

plotBAFLRR *BAF-LRR plot*

Description

Generate a BAF-LRR plot with thorough annotations of canonical positions and data points.

Usage

```
plotBAFLRR(sam.dat, cali.best, oo, origin = "on", para = para)
```

Arguments

sam.dat	numeric matrix, as returned from getSeg() or getSegChr()
cali.best	a list returned from getSumDist()
oo	Origin Cluster, as returned from getOrigin()
origin	if 'on', Origin Cluster will be highlighted on the plot.
para	list, parameters returned from getPara()

Value

NULL

Author(s)

Bo Li

Examples

```
data(A0SD.BAF)
data(A0SD.LRR)
seg.dat=c()
for(CHR in c(8,9,10)){
baf=A0SD.BAF[A0SD.BAF[,2]==CHR,]
lrr=A0SD.LRR[A0SD.LRR[,2]==CHR,]
x=getSegChr(baf,lrr)
seg.dat=rbind(seg.dat,x)
}
dd.dat=seg.dat[,2:8]
rownames(dd.dat)=seg.dat[,1]
mode(dd.dat)='numeric'
para=getPara()
oo=getOrigin(dd.dat,para=para)
cali=getSampleAGP(dd.dat,oo,para=para)
plotBAFLRR(dd.dat,cali,oo,para=para)
```

`plotIdentifiableZone` *Plot Identifiable Zone on sAGP-SAF plot*

Description

This function generates diagnostic identifiable zones on sAGP-SAF plot.

Usage

```
plotIdentifiableZone(nt, nb,add=FALSE,legend=TRUE,title=TRUE)
```

Arguments

`nt`	integer, number of total alleles.
`nb`	integer, number of minor allele.
`add`	logic, if TRUE, areas of identifiable zones will be overlaid onto the current active device.
`legend`	logic, if TRUE, legend indicating scenario A1, A2, B and C will be added to the plot.
`title`	logic, if TRUE, values of nt and nb will be titled to the plot.

Details

To estimate Cancer Cell Fraction using two known quantities: segment-specific AGP (sAGP) of the DNA segment that mutation occurred on and somatic allele frequency (SAF) of the mutation, it is helpful to learn how different temporal order scenarios affect the inference. This function places the uniquely identifiable regions as well overlapping, unidentifiable regions on sAGP-SAF plot to visually diagnose how well the four temporal scenarios can be separated.

Value

NULL

Author(s)

Bo Li

Examples

```
par(mfrow=c(1,3),mar=c(4,4,1,0))
## hemizygous deletion
plotIdentifiableZone(1,0)
## cn-LOH
plotIdentifiableZone(2,0)
## hemizygous amplification
plotIdentifiableZone(3,1)
```

prior *Prior parameters for MCMC*

Description

A pre-selected set of prior parameters required to run `DPdensity()`

Usage

```
data(prior)
```

Format

A list containing 7 parameters. $ a0 : real 0.1 $ b0 : real 0.1 $ m1: real 0.5 $ psiinv1: real 0.01 $ nu1: real 4 $ tau1: real 1 $ tau2: real 100

SampleNMM *Normal-Uniform mixture model fitting*

Description

This function fits the distribution of a input vector with mixture of Normal and Uniform distributions and returns the BIC score for the model.

Usage

```
SampleNMM(Y)
```

Arguments

`Y`	input vector. In the context of subclonality inference, `Y` can either be sAGP or CCF values from one sample.

Details

The model is: Y~ A*U(0,1)+(1-A)*Normal(mu,sigma), where A, mu and sigma are parameters to be fitted.

Value

a list containing the following elements:

`BIC`	BIC score from the model.
`fit`	a vector containing three numbers, in the order of fitted A, mu and sigma.

Author(s)

Bo Li

Examples

```
Y=c(rnorm(100,0.5,0.3),runif(100,0,1))
SampleNMM(Y)
```

Index