Package ‘cancerTiming’

October 12, 2022

Title Estimation of Temporal Ordering of Cancer Abnormalities
Version 3.1.8
Date 2016-04-01
Author Elizabeth Purdom
Maintainer Elizabeth Purdom <epurdom@stat.berkeley.edu>
Description Timing copy number changes using estimates of mutational allele frequency from resequencing of tumor samples.
Depends R (>= 2.11.0)
Suggests GenomicRanges, IRanges, S4Vectors
Imports utils, stats, grDevices, graphics, LearnBayes, gplots
License GPL (>= 2)
NeedsCompilation no
Repository CRAN
Date/Publication 2016-04-03 00:22:46

R topics documented:

- alleleFrequency ................................................................. 2
- bootstrapEventTiming ....................................................... 3
- eventTiming ................................................................. 4
- eventTimingOverList ....................................................... 7
- hg19chromosomes ......................................................... 8
- labelSeg ................................................................. 9
- makeEventHistory ....................................................... 10
- mleAF ............................................................. 11
- multidensity ............................................................. 12
- mut2Seg ............................................................. 13
- mutData ............................................................. 14
- plotAlleleByPosition .................................................... 15
- plotAlleleDensity ....................................................... 17
- plotCopies ............................................................. 18
- plotPi0 ............................................................. 19
**alleleFrequency**

Get allele frequencies for tumor data

**Description**

Calculate allele frequencies that would be possible for a variant found in a region with a given number of total copies of the tumor, adjusted for the given percentage of normal contamination.

**Usage**

```r
allAF(totalCopy, normCont = 0, totalCopyNormal = 2,
type = c("mutation", "SNPHet", "SNPHomo"))
contAF(trueAF, totalCopy, normCont = 0, totalCopyNormal = 2,
  type = c("mutation", "SNPHet", "SNPHomo"))
decontAF(contAF, totalCopy, normCont = 0, totalCopyNormal = 2,
  type = c("mutation", "SNPHet", "SNPHomo"))
errorAF(trueAF, seqError = 0)
```

**Arguments**

- `totalCopy`: total number of copies of DNA of the tumor, integer
- `normCont`: proportion of normal contamination (see Details)
- `totalCopyNormal`: total number of copies of DNA in the normal sample
- `type`: type of allele frequency (see Details)
- `trueAF`: vector of true AF values in tumor
- `contAF`: vector of observed (contaminated by normal) allele frequency values
- `seqError`: The probability of sequencing error per base

**Details**

`contAF` takes as input a true allele frequency and then adjusts it based on the given normal contamination, to give the allele frequencies actually expected from the mixture of tumor and normal cells present. `decontAF` does the reverse, taking contaminated AF and calculating the true AF that is implied for the given normal contamination.

`errorAF` gives the adjusted allele frequencies to account for sequencing error.

`allAF` returns all the possible allele frequencies for a variant, namely (0:totalCopy)/totalCopy. For type="mutation", only alleles from (1:totalCopy)/totalCopy are calculated, since the assumption is that mutated (relative to germline) regions by definition are not expected to be present at 0 copies.
in the tumor. For ‘allAF’, ‘normCont’ can be a vector, in which case all the allele frequencies are calculated for all the values of ‘normCont’; for the remaining functions, it must be a single value.

Only integer values of ‘totalCopy’ are allowed.

‘type’ refers to the type of variant, relative to the normal. Namely, how many copies of the variant are in the normal. Therefore ‘mutation’ implies 0 copies in the normal, ‘SNPHet’ implies 1 copy in the normal, and ‘SNPHomo’ implies variant is in every copy of the normal (which could be not equal to 2 if totalCopyNormal is not 2).

For the functions seqError, contAF, and decontAF, ‘totalCopy’, ‘normCont’, ‘totalCopyNormal’, and ‘seqError’ can be vectors of the same length as the input.

Value

For ‘allAF’, a list of the same length of ‘normCont’. Each element of the list is a vector of the entire set of possible allele frequencies for the corresponding level of normal contamination.

For ‘contAF’, ‘decontAF’, and ‘seqError’ a vector the same length as the inputted allele frequencies that were to be converted.

Author(s)

Elizabeth Purdom

Examples

```r
contAF(1/3, totalCopy=3, normCont=.1, type="mutation")
contAF(c(.5,.5,.5), totalCopy=c(2,4,6), normCont=.2)

#allele frequencies possible for a location that is same
#as reference in the normal, for three different values of normal contamination
allAF(totalCopy=3, normCont=c(0,.1,.5), type="mutation")
#same, but for those that are heterozygous in the normal
allAF(totalCopy=3, normCont=c(0,.1,.5), type="SNPHet")
```

---

**bootstrapEventTiming**  
*Bootstrap the results of eventTiming*

**Description**

Create bootstrap estimates of pi base on the results of a call to eventTiming.

**Usage**

```r
bootstrapEventTiming(eventOrdering, B, type = c("parametric", "nonparametric"), pi, x, m, call)
```
Arguments

- **eventOrdering**: output from eventTiming
- **B**: number of bootstrap samples to take
- **type**: type of bootstrap confidence interval to calculate, one of “parametric”, “non-parametric”
- **pi**: the estimate of $\pi$ from which to create bootstrap samples (if type=“parametric”). If not given, will use output from eventOrdering
- **x**: vector. the number of reads/fragments containing the variant; not needed if output from eventOrdering given.
- **m**: vector. the number of reads/fragments covering the location with the variant (the coverage); not needed if output from eventOrdering given.
- **call**: information about the call given to original eventTiming command. Only needed if eventOrdering object not given (mainly used for internal calls within eventTiming function)

Value

Matrix of dimension (B,length(pi)) with the estimate of pi for each of the bootstrap samples

Author(s)

Elizabeth Purdom

Examples

```r
## can do this within eventTiming function, but here is an example doing it separately...
data(mutData)
ACNLOH<-matrix(c(1,3,1,0),ncol=2,nrow=2,byrow=TRUE)
onlyMuts<-subset(mutData,is.na(rsID) & position <= 1.8E7)
onlyMuts$t_depth<-onlyMuts$t_ref_count+onlyMuts$t_alt_count
x<-eventTiming(x=onlyMuts$t_alt_count,m=onlyMuts$t_depth,
               history=ACNLOH,totalCopy=2,type=“CNLOH”,normCont=0.22,returnAssignments=TRUE)
piBoot<-bootstrapEventTiming(x,B=100,type=“parametric”)
```

**eventTiming**

*Estimate the time of events in tumor data*

Description

Estimate the proportion of time spent between different chromosomal abnormalities based on the allele frequencies of mutated locations.
Usage

eventTiming (x, m, history, totalCopy,
method = c("fullMLE", "partialMLE", "Bayes"),
type = c("gain", "CNLOH"), seqError = 0, bootstrapCI = NULL,
B = if (method == "Bayes") 10000 else 500, CILevel = 0.95, normCont = 0,
verbose = TRUE, returnAssignments = FALSE, coverageCutoff = 1,
minMutations = 10, init = NULL, maxiter = 100, tol = 1e-04,
mutationId = 1:length(x), ...)

Arguments

x vector. the number of reads/fragments containing the variant
m vector. the number of reads/fragments covering the location with the variant (the coverage)

history a matrix, based on the history of the region (see Details)
totalCopy integer. the total number of copies of the tumor DNA for this region

method what estimation method to use, one of “fullMLE”, “partialMLE”, “Bayes”
type type of region, either a gain or a CNLOH region

seqError Probability of sequencing error

bootstrapCI type of bootstrap confidence interval to calculate, one of “parametric”, “non-parametric”. If NULL, then the confidence interval is not calculated

B number of bootstrap samples to take (or simulations from the posterior for Bayesian estimation)

CILevel At what level the confidence intervals should be calculated.

testCont the proportion of normal contamination, between 0 and 1.

verbose logical. Whether to give additional warnings as the program is running.

returnAssignments logical. Whether to return the probabilistic assignments of mutations to allele frequencies generated by the EM algorithm. Also returns the x,m values for those that pass the filter.

coverageCutoff minimum value for m[i]; any entries with m[i]<coverageCutoff are ignored in estimation.

minMutations minimum number of mutations required.

init initial value of multinomial parameter q passed to estimateQ.

maxiter maximum number of iterations in calculation q.

tol tolerance in the convergence of q

mutationId identification values of the mutations (vector of the same length as x and m). Default is indexing values, 1:length(x). Used when returnAssignments=TRUE so that the assignments of the mutations to allele frequencies can be linked with the original mutations if there has been filtering in eventTiming, e.g. due to depth of coverage.
Arguments passed to internal fitting function for Bayesian Method. For example, ‘alpha’ gives the Dirichlet prior of the bayesian estimate (default=1), ‘tdf’ gives the number of degrees of freedom for the t proposal density used in the bayesian estimate (default=4), ‘bayesApproxMethod’ gives the method for calculating the approximate distribution (default is “sir”; “inv” is for K=1 when the problem is 1-dimensional and one can easily grid and get the approximate posterior density and cdf).

Value

A list with values

<table>
<thead>
<tr>
<th>Argument</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>pi</td>
<td>estimate of pi vector</td>
</tr>
<tr>
<td>piCI</td>
<td>bootstrap confidence interval, if requested</td>
</tr>
<tr>
<td>q</td>
<td>estimate of the multinomial parameter q</td>
</tr>
<tr>
<td>perLocationProb</td>
<td>output from estimateQ giving per location P(P[i]</td>
</tr>
<tr>
<td>optimDetails</td>
<td>optimization details from estimateQ</td>
</tr>
<tr>
<td>call</td>
<td>list of the parameters of the call to the function: history, totalCopy, type, exactAllele, normCont, coverageCutoff, minMutations. In addition, ‘alleleSet’ is included in this list, which is the set of possible alleles for this history, after adjusting for normal contamination.</td>
</tr>
</tbody>
</table>

Author(s)

Elizabeth Purdom

References


Examples

data(mutData)
ACNLOH<-matrix(c(1,3,1,0),ncol=2,nrow=2,byrow=TRUE)
onlyMuts<-subset(mutData,is.na(rsID) & position <= 1.8E7)
onlyMuts$t_depth<-onlyMuts$t_ref_count+onlyMuts$t_alt_count
x<-eventTiming(x=onlyMuts$t_alt_count,m=onlyMuts$t_depth,
    history=ACNLOH,totalCopy=2,type="CNLOH",normCont=0.22)
x$pi #estimate of time of stages
x$q #estimate of the multinomial (likelihood of each of the alleles)
x$call$alleleSet #possible set of alleles after #adjusting for normal contamination
**eventTimingOverList**  

EventTimingOverList is a wrapper to EventTiming that allows for timing of common events over several regions of a sample and/or multiple samples. GetPi0Summary gets summary information about \( \pi_0 \) (the first event) from the output of that function and returns a simple dataframe of the estimate of \( \pi_0 \) for every region and sample combination.

### Usage

```r
eventTimingOverList(dfList, normCont, eventArgs)
getPi0Summary(eventList, CI = TRUE)
```

### Arguments

- **dfList**  
  A list of mutation data per sample. Each element of the list corresponds to a sample, and consists of a data frame of mutation information for all of the regions that are to be timed. The data frame must have a certain format, see description below.

- **normCont**  
  A vector of equal length as dfList giving the normal contamination of each sample.

- **eventArgs**  
  List of arguments passed to EventTiming via `do.call`. Should NOT contain the arguments 'x', 'm', 'history', 'totalCopy', 'type', 'mutationId' or 'normCont'.

- **eventList**  
  Output of EventTimingOverList (see below).

- **CI**  
  Logical, whether to grab CI from the output.

### Details

The data frame of mutation data for each sample must have the following columns: 'segId', 'type', 'nMutAllele', 'nReads', 'mutationId'. The rows of the data frame correspond to mutated locations within the sample. 'segId' is an identifier of the segmented region that the mutation is in; EventTiming will be run using the mutation data for each 'segId' in the sample. 'type' identifies the type of segment, one of c("Other","CNLOH","SingleGain","Diploid","DoubleGain"). 'nMutAllele' gives for every row (mutated location) the number of reads with the mutation and corresponds to 'x' imputed into EventTiming. 'nReads' gives the total number of reads covering the location and corresponds to 'm' imputed into EventTiming. 'mutationId' is a unique identifier for the mutation.

For arguments passed to EventTiming, it is recommended to pass verbose=FALSE if you want to avoid many warnings about mutations that do not meet the necessary criteria of coverage, etc.

### Value

EventTimingOverList returns a list equal to the number of samples. Each element of that list (i.e. per sample) is itself a list of length three corresponding to the three types of regions that can be timed, "CNLOH", "SingleGain", and "DoubleGain". Each of these gives the output of EventTiming per region of that type.
hg19chromosomes

Definitions of the p,q, and centromeres of chromosomes

Description

Gives the coordinates of the p.q, and centromeres of human chromosomes, based on hg19

Usage

hg19chromosomes

Format

A data.frame containing 72 rows and 5 columns.

<table>
<thead>
<tr>
<th>chr</th>
<th>chromosome</th>
</tr>
</thead>
<tbody>
<tr>
<td>start</td>
<td>start position</td>
</tr>
<tr>
<td>end</td>
<td>end position</td>
</tr>
<tr>
<td>label</td>
<td>one of “p”, “q”, and “centromere”</td>
</tr>
<tr>
<td>width</td>
<td>width of region</td>
</tr>
</tbody>
</table>
**Description**

Helper functions for plotting and working with chromosomes and segmentations

**Usage**

```r
labelSeg(chr, start, end, pctOv = 0.1)
numChromosome(chr)
divideGenome(size=10)
```

**Arguments**

- `chr`: chromosome
- `start`: start position of segment
- `end`: end position of segment
- `pctOv`: required amount of percent overlap needed in order to report
- `size`: Size, in MB, of the desired equally spaced divisions of the genome

**Details**

`labelSeg` creates labels 'p', 'q', and 'pq' for segmentations based on the overlap of the segment with the p/q portions of the human chromosome (hg19). It uses the data stored in hg19chromosomes with the package (under the directory 'extdata', not as a dataset of the package). Uses the `findOverlaps` function of GenomicRanges.

`numChromosome` takes character valued chromosome labels ("1","2","X","Y") and converts them into values 1:23. This is useful for sorting chromosome values, for example.

`divideGenome` makes segments of the genome of size equal to 'size' times 1e6, i.e. size-MB length intervals, also based on the hg19chromosomes data described above.

**Value**

vector of numerical values between 1:23

**Author(s)**

Elizabeth Purdom

**See Also**

hg19chromosomes, findOverlaps
**Examples**

```r
data(mutData)
segData<-data.frame(chromosome="17", start=c(0,1.8e7+1),
end=c(1.8e7,max(mutData$position)),
totalCpy=c(2,NA),
markRegion=c(FALSE,TRUE))
if(require(GenomicRanges) & require(IRanges)) labelSeg(chr=segData$chromosome,
start=segData$start,end=segData$end)
chr<-c("1","4","10","23","X")
chr[order(chr)]
chr[order(numChromosome(chr))]
```

---

**Description**

Create the event history matrix needed for the event timing algorithm

**Usage**

```r
makeEventHistory(type = c("gain", "LOH"), copies = NULL,
totalCopy = sum(copies), onlyIdentifiable = TRUE)
```

**Arguments**

- **type**
- **copies** a vector of length 2, giving the number of copies of the maternal and paternal chromosomes
- **totalCopy** an integer between 2 and 5. totalCopy must be the sum of the numbers given for the argument copies
- **onlyIdentifiable** logical. Only return event matrices corresponding to event histories with identifiable π0?

**Value**

The A matrix relates allele frequencies to the event history. The vector $A\pi$ gives the probabilities of each allele frequency possible for the specified event.

**Author(s)**

Elizabeth Purdom

**References**

Purdom, E, et al. (submitted). “Timing Chromosomal Abnormalities within Cancer Samples.”
mleAF

Examples

    makeEventHistory(type="gain",totalCopy=8)
    makeEventHistory(type="gain",copies=c(1,4),totalCopy=5,onlyIdentifiable=FALSE)

---

mleAF Estimate the most likely allele frequency

Description

Estimate the number of copies a mutation is found in, based on which allele value maximizes the
binomial likelihood after correcting for normal contamination and seqError.

Usage

mleAF(x, m, totalCopy, maxCopy=totalCopy, seqError = 0, normCont = 0)

Arguments

- **x**: vector. the number of reads/fragments containing the variant
- **m**: vector. the number of reads/fragments covering the location with the variant (the
  coverage)
- **totalCopy**: The total number of copies (maternal and paternal combined), can be vector with
  length equal to length(x)
- **maxCopy**: The maximum number of copies of either maternal or paternal alleles, can be
  vector with length equal to length(x)
- **seqError**: The probability of sequencing error per base, can be vector with length equal to
  length(x)
- **normCont**: Percentage of normal contamination, can be vector with length equal to length(x)

Details

maxCopy and totalCopy are used to determine the possible allele frequencies in a pure tumor cell,
given by 1:maxCopy/totalCopy. The default of maxCopy=totalCopy ensures that all theoretically
possible alleles are considered given the lack of further information, but in general will not be
accurate. For example, if the region has allelic copy 2/3, then there are only three possible allele
frequencies rather than five.

Value

List with following values:

- **perLocationProb**: matrix of dimension (number of locations) x (number of possible allele frequen-
cies), with each row corresponding to a given location and each column giving
the probability of observing the data for that location for each of the possible allele frequencies
assignments data.frame of dimension (number of locations) x 3, with columns ncopies=estimate of number of copies mutation is found in, based on which maximizes the likelihood, totalCopy=totalCopy given by user, AF=estimate of true allele frequency given by ncopies/totalCopy

alleleSet Only returned if the parameters totalCopy, maxCopy, seqError, and normCont are of length=1. A data.frame with rows equal to number of possible alleles and three columns, tumorAF=the allele frequency in the pure tumor, AF= the corresponding allele frequency after adjusting for normal contamination and sequencing error, frequency = number of locations with that allele frequency.

Author(s)
Elizabeth Purdom

References

Examples
#example of CNLOH
m<-c(24,41,40,15)
x<-c(13,21,17,12)
nc<-c(0.27,0.39,0.49,0.22)
mleAF(x=x,m=m,totalCopy=2,maxCopy=2,normCont=nc)
mleAF(x=x,m=m,totalCopy=c(2,3,2,3),maxCopy=2,normCont=nc)
#note the difference in output if instead all data is from
#same sample (shares normal Contamination estimate)
mleAF(x=x,m=m,totalCopy=2,maxCopy=2,normCont=nc[1])

multidensity plot multiple density functions on top of each other

Description
plots multiple density functions on top of each other, calculating the correct xlim, ylim, etc.

Usage
multidensity(x, col = palette(), lwd = 1, lty = 1, xlab, ylab = "Density", ...)

Arguments
x list of values to create density from (can be a data.frame which where columns are elements of the list)
col colors for each density plot
lwd lwd for each density plot
mut2Seg

ltty for each density plot
xlim user defined xlim
ylab user defined ylab
... plotting parameters passed to initial plot command.

Author(s)

Elizabeth Purdom

Examples

```r
x<-lapply(c(1,2,3),function(x){rnorm(100,mean=x)})
multidensity(x)
```

mut2Seg Align mutations to segments

Description

A function to match mutations into the segments that include them.

Usage

```r
mut2Seg(mutData, segData, verbose = TRUE)
```

Arguments

mutData matrix or data.frame. Column names must include ‘chr’ and ‘position’
segData matrix or data.frame. Column names must include ‘chr’, ‘start’, and ‘end’ (in any order)
verbose logical. If TRUE gives information about the progress and possible problems (e.g. if different chromosome names in the two sets)

Details

This function finds which segments in the segData file contain the mutations and returns a data.frame with the concatenation of the mutation information and the segments information. The function relies on findOverlaps in the GenomicRanges package in bioconductor.

Value

Data frame with concatenated values.

Author(s)

Elizabeth Purdom
mutData

See Also

findOverlaps

Examples

```r
if(require(GenomicRanges) & require(IRanges)){
data(mutData)
colnames(mutData)[1]<-c("chr")
segs<-data.frame(chr=c(17,17),start=c(1,1.8e7+100),end=c(1.8e7,81195210),
normCont=0.22,segId=c("CNLOH","Other"))
##Trivial segmentation annotation for example
mutId<-mut2Seg(mutData,segs)
head(mutId)
}
```

---

**mutData**

*Example Mutation Data*

**Description**

Example mutation data

**Usage**

data(mutData)

**Format**

A data frame with 1007 observations on the following 10 variables.

- **chromosome** chromosome
- **position** position
- **refbase** reference base at this location
- **mutbase** variant base at this location
- **rsID** dbSNP database number, NA if not in database.
- **t_ref_count** number of fragments in tumor with the reference base
- **t_alt_count** number of fragments in tumor with the variant base
- **allelefreq** observed allele frequency
- **n_ref_count** number of fragments in normal with the ref base
- **n_alt_count** number of fragments in normal with the variant base

**Details**

Mutation data from chr17. The CNLOH region is from positions 0 to 1.8E7. The normal contamination estimate in the paper was given as 0.22.
**plotAlleleByPosition**

**Source**


**Examples**

```r
data(mutData)
head(mutData)
# only mutations in the CNLOH region
onlyMuts<-subset(mutData,is.na(rsID) & position <= 1.8E7)
```

**plotAlleleByPosition**  
*Plot allele frequencies by position*

**Description**

Plot observed allele frequencies from sequencing data against their location on the chromosome.

**Usage**

```r
plotAlleleByPosition(mutData, segmentData = NULL,
whChr = 1:22, chromosomeId = "chr",
sampleName = NULL, sample = FALSE, tumorAFId, positionId, type = "mutation",
startId = "start", endId = "end", segFactorId, tCNId, MarkId, segColors,
normCont = NULL, addData = NULL, addColor="red",col="black",pch=1,lwd=2,
xlim,ylab="Allele Frequency",...)
```

**Arguments**

- `mutData`: data.frame with mutation data set to be plotted
- `segmentData`: (optional) segmentation data
- `whChr`: which chromosome to plot
- `chromosomeId`: column name for chromosome (must be same in all data.frames)
- `sampleName`: id printed on the plot to identify the sample
- `sample`: logical. If true, take only a random sample of 10,000 locations for the chromosome. Can speed up for plotting SNPs.
- `tumorAFId`: column name for the allele frequency in mutData
- `positionId`: column name for the allele location in mutData
- `type`: type of allele frequency plotted (passed to `allAF` in order to create the lines for the expected AF)
- `startId`: column name for the start of the segmentation (in segData)
- `endId`: column name for the end of the segmentation (in segData)
- `segFactorId`: column name for the factor for segmentations (in segData).
plotAlleleByPosition

- **tCNId**: column name that gives the total copy number for the segmentation (in segData); needed if give normCont to calculate expected AF.
- **MarkId**: column name of a column with logical values that identifies segments that should be marked up with hash marks.
- **segColors**: vector of colors for the segmentations. Should be as long as the number of levels of segFactorId.
- **normCont**: percent normal contamination. If missing, then lines for the expected AF will not be calculated.
- **addData**: data frame with another set (example germline SNPs) to be plotted in red.
- **addColor**: color for the additional data.
- **lwd**: line width of the lines for the expected AF.
- **ylab**: label for y-axis.
- **xlim**: xlim boundaries. If missing, will be calculated.
- **col**: col for the mutData points.
- **pch**: pch for the mutData points.
- **...**: arguments passed to initial plotting command.

**Value**

returns invisibly the vector of colors for the segmentations, useful for making legends (see the example)

**Author(s)**

Elizabeth Purdom

**Examples**

data(mutData)

# only mutations in the CNLOH region
onlyMuts<-subset(mutData,is.na(rsID) & position <= 1.8E7)

snps<-subset(mutData,!is.na(rsID) )

segData<-data.frame(chromosome="17",start=c(0,1.8e7+1),
end=c(1.8e7,max(mutData$position)),
totalCpy=c(2,NA),markRegion=c(FALSE,TRUE))

out<-plotAlleleByPosition(onlyMuts,whChr=17, segmentData=segData,
tCNId="totalCpy",normCont=0.22, addData=snps,pch=19,
addColor="grey",MarkId="markRegion",
segColors="pink",xaxt="n",xlab="Position", segFactorId="totalCpy",
chromosomeId = "chromosome",tumorAFId="allelefreq",
positionId="position",type="mutation")
axis(1,line=1,tick=FALSE)

legend("topright",legend=c(names(out),"unknown"),fill=c(out,NA),
title="Total Copy Number")
plotAlleleDensity  

Plot density/histogram of allele frequencies

Description

Plot of densities/histograms of allele frequencies, useful for evaluating normal contamination estimate, total Copy number estimates, etc.

Usage

plotAlleleDensity(af, depth, groupingId, totalCopy, groupCol=palette(), normCont = 0, type="mutation", minDepth = 40, lineCols = c("grey", "tan4"), minMut = 40, histogram = FALSE)

Arguments

af vector of allele frequencies
depth coverage of the location
groupingId grouping variable for allele frequencies (single density curve/histogram for each id)
totalCopy the total copy number for the allele frequencies plotted (must be the same)
groupCol colors for the different groups (if histogram=FALSE)
type type of allele frequency (mutation, SNPHet, SNPHomo), passed to allAF
normCont percentage of normal contamination. Can be vector of different values.
minDepth min required depth in order to include it in the density/histogram
lineCols colors of the vertical lines (each color is for different normal contamination)
minMut minimum number of mutations per group in order to plot the group
histogram logical. should the plot be superimposed density curves (FALSE), or a single histogram per group (TRUE)?

Value

Returns invisibly the data used (i.e. passed minimum cutoff criteria), separated by the groupingId given by the user

Author(s)

Elizabeth Purdom
Examples

data(mutData)
#only mutations in the CNLOH region
onlyMuts<-subset(mutData,is.na(rsID) & position <= 1.8E7)
plotAlleleDensity(onlyMuts$allelefreq,onlyMuts$t_ref_count+onlyMuts$t_alt_count,
totalCopy=2,normCont=c(0,0.22),minMut=0,minDepth=0,hist=TRUE)

gpId<-factor(is.na(mutData$rsID),levels=c("TRUE","FALSE"),labels=c("Mutations","SNPs"))
plotAlleleDensity(mutData$alleleFreq,mutData$t_ref_count+mutData$t_alt_count,
groupCol=c("black","red"),totalCopy=2,groupingId=gpId,minMut=0,minDepth=30,hist=FALSE)
legend("topleft",levels(gpId),fill=c("red","black"))

plotCopies

plot segmentation values against each other

Description

Plot different values per segment against each other (e.g. minor and major allele estimates).

Usage

plotCopies (x, y, nX, nY, xleg, yleg, onlyPositive = TRUE,
    equalAxis = TRUE, integerLegend = TRUE, xlim, ylim, ...)

Arguments

x The values plotted for x coordinates (e.g. segmentation value for minor allele)
y The values plotted for y coordinates (e.g. segmentation value for major allele)
nX The grouping id for the x coordinate (e.g. assignment of number of copies)
nY The grouping id for the y coordinate (e.g. assignment of number of copies)
xleg title for the legend of the x values
yleg title for the legend of the y values
xlim limits for the x axis
ylim limits for the y axis
onlyPositive only plot values positive in x and y
equalAxis xlim and ylim are forced to be the same
integerLegend Only give integers
... passed to the scatter plot command

Author(s)

Elizabeth Purdom
Examples

```r
cp1 <- c(0, 0, 0, 1, 1, 1, 2, 2, 2, 3, 3, 3)
cp2 <- c(0, 1, 2, 2, 2, 2, 3, 3, 3)
seg1 <- jitter(c(0, 0, 0, 1, 1, 1, 2, 2, 2, 3, 3, 3))
seg2 <- jitter(c(0, 1, 2, 2, 2, 2, 3, 3, 3))
plotCopies(x = seg1, y = seg2, nX = cp1, nY = cp2)
```

**plotPi0**

Confidence intervals for $\pi_0$

Description

A plotting function to plot the confidence intervals of the estimated $\pi_0$ values returned from event-Timing

Usage

```r
plotPi0(est, ui, li, samples, labels = NULL, xlab = "Segment",
ylab = expression(pi[0]),
pchFac = rep(1, length(est)), whHighlight = NULL,
nMut = NULL, xorder = NULL, ...)
```

Arguments

- `est` vector of the estimates of $\pi_0$
- `ui` vector of the upper values for the confidence interval of the $\pi_0$
- `li` vector of the lower values for the confidence interval of the $\pi_0$
- `samples` vector indicating which sample each $\pi_0$ corresponds to (will be converted to factor with factor command)
- `labels` character vector of the labels of each estimate (optional)
- `xlab` label for the xaxis, defaults to “Segment”
- `ylab` label for the yaxis, defaults to “$\pi_0$”
- `pchFac` vector of pch values for the center of the confidence interval
- `whHighlight` a vector of indices of the estimates that should be ‘highlighted’ by coloring their confidence intervals red (optional)
- `nMut` a vector of values of the number of mutations from each confidence interval to be printed at the top of the plot
- `xorder` a vector giving an ordering of the estimates for the confidence intervals; if NULL, an order within each sample is created automatically.
- `...` arguments passed to plotCI function

Value

The order of the estimates found internally by the program (or given by the user) are returned invisibly.
Author(s)

Elizabeth Purdom

See Also

plotCI, evenTiming

plotSegmentation

plot segmentation(s) against positions

Description

Plot the values of a segmentation against chromosome position.

Usage

plotSegmentation(segs, valId, col = palette(), lty=1, lwd = 2, xlim, ylim, xlab="Position", ylab = valId,...)

Arguments

segs list of segmentation data.frames. Each data.frame must have ‘start’ and ‘end’ as the column names of the limits of the segmentations.
valId name of the column with the segmentation value to be plotted (must be the same in all data.frames)
col colors for the segmentation (1 per element of the list of segmentations)
lty lty for the lines
lwd lwd for the lines
xlim x limits
ylim y limits
xlab The label for the x-axis
ylab The label for the y-axis
... passed to initial plotting command

Value

returns invisibly the col and lty after any processing done, useful for legends.

Author(s)

Elizabeth Purdom
Examples

data(mutData)
segData<-data.frame(chromosome="17", start=c(0,1.8e7+1),
end=c(1.8e7,max(mutData$position)), val=c(2,3))
cp1<-data.frame(chromosome="17", start=c(0,1.8e7+1),
end=c(1.8e7,max(mutData$position)), val=c(1,1))
cp2<-data.frame(chromosome="17", start=c(0,1.8e7+1),
end=c(1.8e7,max(mutData$position)), val=c(.9,2))
out<-plotSegmentation(list(total=segData,cp1=cp1,cp2=cp2),
valId="val", lwd=2, ylab="Segmentation Value")
legend("topright", names(out),fill=out)

plotSeqCount

Basic function for plotting the ratio of tumor to normal coverage

Description

Basic function for plotting the ratio of tumor to normal coverage, and calculating the average over segments. A simple tool for assessing copy number breaks.

Usage

plotSeqCount(position, t_count, n_count, ylim=NULL, normFac=1,
segs, segColors = palette(), ...)

Arguments

position vector of position of the base
t_count vector of number of reads overlapping position in tumor
n_count vector of number of reads overlapping position in normal
ylim set the limits of y axis. If NULL, default values found.
normFac a normalizing factor to correct for
segs factor as same length as position, identifying which segment a count is from
segColors ... passed to plot

Author(s)

Elizabeth Purdom
readSimulation

Simulate reads based on an event matrix

Description

Simulate reads based on an event matrix

Usage

readSimulation(B, alleleSet, q, totalCopy, mutRate = NULL, seqError = 0, fixedN = FALSE, normCont = 0, aveReadCoverage = 30, countDistribution = NULL)

Arguments

B the number of simulated data sets to make
alleleSet a vector of expected allele frequencies, without contamination or sequencing error accounted for
q integer. the number of expected allele frequencies
totalCopy integer. The total number of copies in the final stage, i.e. at the end of the event
mutRate a number between 0 and 1. The mutation rate, defined as the number of mutations observed divided by number of nucleotides sequenced or examined
seqError a number between 0 and 1 representing the sequencing error. The default value is 0, i.e. no sequencing error.
fixedN logical. Should the number of mutations be fixed? By default set to FALSE. If set to TRUE, the expected number of mutations is used, i.e. mutRate*B
normCont a number between 0 and 1 describing the amount of normal contamination present in the sample. The default value is 0, i.e. no normal contamination.
aveReadCoverage average read coverage of an allele. The default value is 30 reads per allele.
countDistribution optional. Provide an empirical distribution for read coverage. If given, the argument aveReadCoverage is ignored, and the mean of the empirical distribution is used. By default, set to NULL.

Value

Returns a data frame with simulated read counts under the normal contamination, sequencing error, allele frequency distribution, and read count distributions provided.

Author(s)

Elizabeth Purdom
Examples

# simulate from CNLOH event with pi[0]=.1
Amat<-makeEventHistory(totalCopy=2,type="LOH")[[1]]
pVal<-c(.1,.9)
qvec<-prop.table(Amat%*%pVal)

sims<-readSimulation(10, alleleSet=allAF(totalCopy=2)[[1]], q=qvec,
totalCopy=2, mutRate = 100, seqError = 0.1, fixedN = TRUE,
normCont = 0.1, aveReadCoverage = 30)
Index

* datasets
  hg19chromosomes, 8
  mutData, 14

allAF (alleleFrequency), 2
alleleFrequency, 2
bootstrapEventTiming, 3
contAF (alleleFrequency), 2
decontAF (alleleFrequency), 2
divideGenome (labelSeg), 9
errorAF (alleleFrequency), 2
eventTiming, 4, 8
eventTimingOverList, 7
findOverlaps, 9, 14
getPi0Summary (eventTimingOverList), 7
hg19chromosomes, 8, 9
labelSeg, 9
makeEventHistory, 10
mleAF, 11
multidensity, 12
mut2Seg, 13
mutData, 14
numChromosome (labelSeg), 9
plotAlleleByPosition, 15
plotAlleleDensity, 17
plotCopies, 18
plotPi0, 19
plotSegmentation, 20
plotSeqCount, 21
readSimulation, 22