Package ‘Butte’

September 16, 2023

Title BoUnds of Time Till Expansion, inferring the SCNA arrival time in the somatic evolution toward the most recent common ancestor of tumor sample(s).

Version 0.0.0.9000

Description Butte is a computational framework to calculate the arrival time and initiation time of a clonal somatic copy number alteration (SCNA) observed in patient tumor sample(s). Currently the method focuses on copy number gains. A genomic region at an observed clonal SCNA state has evolved during the timeline from the germline to the founder cell of the clonal expansion. The timeline can be divided into three fractions in term of the copy number evolution. The first time fraction (T0) is the SCNA initiation time when the first gain occurs. The third fraction (TK) is the arrival time which measure the delay from the last gain to the start of population expansion. Butte can estimate the initation time and arrival time of an SCNA (with the total copy number as high as 7) given the read counts of somatic single nucleotide variants (SSNVs) occurred within corresponding genomic region and tumor purity. To do so, Butte adopts EM algorithm to find the allele state distribution of SSNVs. Then it either directly solve the time fraction (for SCNAs with identifiable history matrices), or adopts linear programming to calculate the upper bounds of these time fractions if multiple history matrices can exist for the SCNA or the underlying linear system is underdetermined.

License MIT License

Encoding UTF-8

LazyData true

RoxygenNote 7.2.2

R topics documented:

.estimateQ *EM algorithm to calculate q*

Description

Estimate the proportion the proportions of SSNVs at each allele state from sequencing data.

Usage

```
.estimateQ(
  x,
  m,
  alleleSet,
  alleleFreq = NULL,
  history,
  type = c("identifiable", "butte"),
  init = NULL,
  maxiter = 100,
  tol = 1e-04,
  xGreaterZero = TRUE,
  useGradient = FALSE
)
```

Arguments

`x`	vector of number of reads supporting SSNVs
`m`	vector of total read depth for SSNVs
`alleleSet`	vector of possible allele frequencies
`alleleFreq`	the observed allele frequencies
`history`	a list of possible evolutionary history matrices, see also function "cnmutHistory"
`type`	set it to be either "identifiable" or "butte"
`init`	the initial values of vector q (probability of a randomly acquired SSNV having each allele state)
`maxiter`	maximum number of iterations in calculation q
`tol`	the tolerance in the convergence of q
`xGreaterZero`	determines whether the likelihood will account for the fact that only observe SSNVs with mutant read count x[i] > 0

Value

A list of possible matrices

`.lpbounds` *Use linear programming to solve the bounds of TK or T0*

Description

This function estimate the bounds of TK, which is the last time interval in the tumor cell envolution history. Based on different possible history given a copy number ratio, the function makes use of linear programming to minimize and maximize TK. The objective function of the optimization if f(x) = ta, which can be written as [0 0 0 0 0 ... 0 1]T*[t1 t2 t3 ... ta]. The first part of constraints are given by (A-qs')t = 0, where s' refers to the transpose vector calculated by rowSum(A). The second part of constraints is the convexity of time vector t. Each element of t refers to a relative fraction of time. We then combine these two constraints into a single linear system formation. he first constraint directly follows from At/c = q, wherer c is a normalizing constant given by the product s'*t.

Usage

```
.lpbounds(q, possible_histories, scost = 100, p0 = FALSE)
```

Arguments

`q` q estimated from data

`possible_histories` matrices of possible SCNA-SSNV histories, see also function "cnmutHistory"

`scost` the cost for slack variables (default 100)

`p0` logical, if TRUE, the upper bounds for T0 will be estimated (instead of TK)

Details

Then the feasibility of the solution region of the linear programming problem will only be influenced by q, which is optimized previously from data. The uncertainty of q can make the solution space infeasible. So we add some slack variables to elasticize the linear programming problem. For details, please check: http://web.mit.edu/lpsolve/doc/Infeasible.htm This elasticizing method will find the approximate bounds of TK close to the constraints. "scost" is the argument adjusting the penalty of the additional slack variables.

Value

the lower and upper bounds of the time duration for the last stage

`bootstrapButte` *Bootstrap for the CI of timing estimates*

Description

the corresponding SCNA evolution.

Usage

```
bootstrapButte(
  eventOrdering,
  B,
  type = c("parametric", "bootstrap"),
  pi,
  x,
  m,
  call
)
```

Arguments

`eventOrdering`
is the list returned by Butte

`B` is the number of bootstrapping samples

`type` "bootstrap" for resample the data; or "parametric" to simulate randomized data

`pi` is the estimate of pi you want to use for bootstrapping

`x` the data to bootstrap

`m` the data to bootstrap

`call` list of original argument settings used in calling Butte

Value

timing estimates of bootstrapped data

`Butte` *Calculate timing of SCNA*

Description

This function estimates the timing of SCNAs for the following two scenarios. For 2:0 3:1 3:0 4:1, where an identifiable history matrix is available, this function returns the time estimate and confidence interval for each stage For SCNAs with multiple or non-indentifible matrix, this function instead returns the bounds (lower and upper) of time period for the last stage of the corresponding SCNA evolution.

Usage

```
Butte(
  x,
  m,
  history,
  nt,
  nb,
  qmethod = c("fullMLE", "partialMLE"),
  type = c("identifiable", "butte"),
  seqError = 0,
  bootstrapCI = NULL,
  B = 500,
  CILevel = 0.9,
  purity = 1,
  verbose = TRUE,
  returnAssignments = TRUE,
  minMutations = 10,
  init = NULL,
  maxiter = 100,
  tol = 1e-04,
  mutationId = 1:length(x),
  ...
)
```

Arguments

`x` vector of number of reads supporting SSNVs

`m` vector of total read depth for SSNVs

`history` a list of possible evolutionary history matrices, see also function "cnmutHistory"

`nt` total copy number

`nb` copy number of the minor allele

`qmethod` suggest to use fullMLE, which is more accurate

`type` set it to be either "identifiable" or "butte", or leave it unset

`seqError` sequencing errors

`bootstrapCI` set to "bootstrap" if non-parametric; or "parametric". This specify the confidence interval method. NULL if do not want to calculate CI

`B` the number of bootstrapping samples

`CILevel` confidence interval level

`purity` tumor sample purity

`verbose` logical. Turn on/off additional warnings.

`returnAssignments`
logical. Whether to return the probabilistic assignments of SSNVs to allele states generated by the EM algorithm, as well as the SSNV read counts (total depth and depth of the mutant allele).

`minMutations` minimum number of SSNVs required for timing analysis

`init` the initial value of vector q passed to function .estimateQ

`maxiter` maximum number of iterations in EM algorithm when calculating the allele state distribution

`tol` the tolerance in the convergence of q

`mutationId` a vector of mutation IDs. The default is 1:length of x.

Value

A list of possible matrices

`cnmutData` *Generate the SCNA SSNV input for running butte*

Description

Four elements will be generated in the output list merged CNA segmentation; SSNV data.frame; cnvHits (index in CNA file, overlapping ssnv) snvHits (index in SSNV file, overlapping with cnv)

Usage

```
cnmutData(scnaFile, ssnvFile, skipchunk = 19)
```

Arguments

`scnaFile` the SCNA segmentation file

`ssnvFile` the SSNV file

`skipchunk` segments with number of data points (probes) no more than this number will be skipped

Value

list of data input for running butte

`cnmutHistory` *generate history matrix in relating SCNA to SSNVs*

Description

Given a SCNA configuration Nt (total copy) and Nb (minor copy) this function produces the possible history matrices in relating CN timing (time period for each stage) to the burden of SSNVs at distinct allele states.

Usage

```
cnmutHistory(nt, nb)
```

Arguments

`nt`	total copy number
`nb`	copy number of the minor allele

Value

A list of possible matrices

`mergeCNA` *Merge the CNA by jumping (neglecting) small segments*

Description

When CNA segmentation contains many small segments, one may want to merge the two neighboring segments by skipping the small segment.

Usage

```
mergeCNA(cnFile, skipchunk = 19, correctMale = FALSE)
```

Arguments

`cnFile`	the SCNA segmentation file
`skipchunk`	segments with number of data points (probes) no more than this number will be skipped
`correctMale`	logical, whether or not divide by 2 for the X chromosome (testing)

Value

data frame of the merged CNA segmentation

scnaInput *Reading and sorting scnaFile*

Description

Reading and sorting scnaFile

Usage

```
scnaInput(scnaFile, skipchunk = 19)
```

Arguments

scnaFile	the SCNA segmentation file
skipchunk	segments with number of data points (probes) no more than this number will be skipped

Value

sorted scna segmentation data frame

scnaTiming *estimate the initiation and arrival time for each SCNA segment given a file containing multiple SCNA segments, and a file containing SSNV data.*

Description

estimate the initiation and arrival time for each SCNA segment given a file containing multiple SCNA segments, and a file containing SSNV data.

Usage

```
scnaTiming(
  scnaFile,
  ssnvFile,
  sn,
  outname,
  public = FALSE,
  pubOrSub = "pubOrSub",
  skipchunk = 19,
  mmut = 10,
  qmethod = "fullMLE",
  B = 100
)
```

Arguments

`scnaFile`	the SCNA segmentation file
`ssnvFile`	the SSNV file
`sn`	sample name
`outname`	output timing file name (including path)
`public`	somatic timeline is focused on public mutations (across multi-samples) or not (otherwise, clonal variants in the specific sample)
`pubOrSub`	the colname for column indicating if the mutation is public or not
`skipchunk`	segments with number of data points (probes) no more than this number will be skipped, set it to zero if not skipped
`mmut`	minimum number of mutations for running timing analysis
`qmethod`	the method for estimating q (probabilities of a randomly acquired mutation having allele state of aj/Nt)
`B`	number of bootstrap for calculating confidence interval

Value

list: timing result; timing table (for visualization) and merged CNA data frame. For butte cases (non-identifiable), pi[1] is the lower bound, and pi[2] is the upper bound. piCI[1,] and piCI[2,] are the bootstrapped confidence interval for the two bounds, respectively.

`ssnvInput` *Reading and sorting ssnvFile*

Description

Reading and sorting ssnvFile

Usage

```
ssnvInput(snvFile)
```

Arguments

`ssnvFile`	the SSNV file

Value

sorted ssnv data frame

vafEst *Return the allele frequencies of SSNVs for each allele state*

Description

Given a SCNA configuration Nt (total copy) and Nb (minor copy), as well as the tumor purity and prevalence of the SCNA, this function returns the expected allele frequencies for possible allele states.

Usage

```
vafEst(ncmut = "all", nt, nb, pu, pa = 1)
```

Arguments

`ncmut`	the number of copies for the mutation, default print for "all"
`nt`	total copy number
`nb`	copy number of minor allele
`pu`	tumor purity
`pa`	prevalence of the SCNA WITHIN the tumor content, default 1 (clonal)

Value

a named vector of expected allele frequency at each allele state

Index