Package ‘seqCBS’

February 20, 2015

Type  Package
Title  CN Profiling using Sequencing and CBS
Version 1.2
Date  2010-05-18
Author Jeremy J. Shen, Nancy R. Zhang
Maintainer Jeremy J. Shen <jeremyjshen@gmail.com>
Description This is a method for DNA Copy Number Profiling using Next-Generation Sequencing. It has new model and test statistics based on non-homogeneous Poisson Processes with change point models. It uses an adaptation of Circular Binary Segmentation. Also included are methods for point-wise Bayesian Confidence Interval and model selection method for the change-point model. A case and a control sample reads (normal and tumor) are required.
License GPL-2
LazyLoad yes
Depends R (>= 2.10), clue
Repository CRAN
Repository/R-Forge/Project seqcnv
Repository/R-Forge/Revision 10
Date/Publication 2012-10-29 08:59:40
NeedsCompilation yes

R topics documented:

seqCBS-package ........................................... 2
BayesCptCI ............................................... 3
CombineCaseControlC .................................. 4
CombineReadsAcrossRuns ............................... 5
getAutoGridSize ......................................... 6
getCountsInWindow ....................................... 7
## seqCBS-package

*Scan Statistics CNV detection using sequencing data*

### Description

CNV detection using matched case-control sequencing read data. It gives a number of scan statistics for the detection of rate differences between two non-homogeneous Poisson Processes, and modified BIC model selection.

### Details

<table>
<thead>
<tr>
<th>Package:</th>
<th>seqCBS</th>
</tr>
</thead>
<tbody>
<tr>
<td>Type:</td>
<td>Package</td>
</tr>
<tr>
<td>Version:</td>
<td>1.2</td>
</tr>
<tr>
<td>Date:</td>
<td>2011-05-18</td>
</tr>
<tr>
<td>License:</td>
<td>GPL-2</td>
</tr>
<tr>
<td>LazyLoad:</td>
<td>yes</td>
</tr>
</tbody>
</table>

---

**Index**

<table>
<thead>
<tr>
<th>Function</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>hppSimulate</td>
<td>8</td>
</tr>
<tr>
<td>JSSim_Meta</td>
<td>8</td>
</tr>
<tr>
<td>JSSim_NormalSim1</td>
<td>9</td>
</tr>
<tr>
<td>JSSim_NormalSim2</td>
<td>9</td>
</tr>
<tr>
<td>JSSim_SpikeMat</td>
<td>9</td>
</tr>
<tr>
<td>JSSim_TumorSim1</td>
<td>10</td>
</tr>
<tr>
<td>JSSim_TumorSim2</td>
<td>10</td>
</tr>
<tr>
<td>nhppRateEstimate</td>
<td>10</td>
</tr>
<tr>
<td>nhppSimConstWindowAnalysis</td>
<td>11</td>
</tr>
<tr>
<td>nhppSimConstWindowGen</td>
<td>13</td>
</tr>
<tr>
<td>nhppSimulate</td>
<td>14</td>
</tr>
<tr>
<td>nhppSpike</td>
<td>15</td>
</tr>
<tr>
<td>nhppSpikeConstWindow</td>
<td>16</td>
</tr>
<tr>
<td>readInput</td>
<td>17</td>
</tr>
<tr>
<td>readListInputFile</td>
<td>18</td>
</tr>
<tr>
<td>readSeq</td>
<td>19</td>
</tr>
<tr>
<td>readSeqChiang</td>
<td>20</td>
</tr>
<tr>
<td>readSeqELANDPaired</td>
<td>21</td>
</tr>
<tr>
<td>relCNComp</td>
<td>22</td>
</tr>
<tr>
<td>ScanBIC</td>
<td>22</td>
</tr>
<tr>
<td>ScanCBS</td>
<td>23</td>
</tr>
<tr>
<td>ScanCBSPlot</td>
<td>25</td>
</tr>
<tr>
<td>ScanCBSimPlot</td>
<td>26</td>
</tr>
<tr>
<td>ScanIterateGrid</td>
<td>27</td>
</tr>
<tr>
<td>ScanStatNewComp</td>
<td>28</td>
</tr>
<tr>
<td>ScanStatRefineComp</td>
<td>29</td>
</tr>
<tr>
<td>SegSeqResProcess</td>
<td>30</td>
</tr>
<tr>
<td>seqCBS_Mpackage</td>
<td>31</td>
</tr>
</tbody>
</table>
BayesCptCI

An overview of how to use the package, including the most important functions

Author(s)

Jeremy J. Shen
Nancy R. Zhang

Maintainer: Jeremy J. Shen <jqshen@stanford.edu> ~ The author and/or maintainer of the package

---

BayesCptCI | Bayesian Point-wise Confidence Interval for Change-Point Model

Description

This algorithm computes a point-wise Bayesian CI for the p parameter in change-point model on binomial process.

Usage

BayesCptCI(cases, controls, CBSRes, stepSize="adaptive", adaptMaxMix=80, alpha=0.05, epsilon=10^-4, epsCDF, verbose=false)

Arguments

cases | A numeric vector of the case/tumor reads
controls | A numeric vector of the control/normal reads
CBSRes | Output from the ScanCBS algorithm on this case/control data
stepSize | An actual point-wise computation is time-consuming; by using stepSize = n, a Bayesian CI is computed at every n reads. The adaptive option gives good computational speed by choosing stepSize based on the data.
adaptMaxMix | An upper bound for the number of unique weights calculated at each change point under the adaptive method. The default is 80 for an average of approximately 5000 mixture components at each point.
alpha | Defaults to 0.05 for the usual CI.
epsilon | The cutoff for the likelihood ratio of a model with shifted change point compared to the ScanCBS estimated change-point. The likelihood decreases exponentially around the true point or a 'good' estimate of it. Only alternatives with above the cutoff likelihood ratio are considered plausible and integrated in following computations.
epsCDF | This is an error tolerance value for finding the quantile of the posterior, which is a Beta mixture distribution.
verbose | If TRUE, then will print much information on each segmentation. For diagnostics only.
Details
This method is a Bayesian point-wise CI for our change-point method. It takes model complexity (number of change points) as a given. With the ScanCBS-estimated change points, it evaluates alternatives for the change points around the estimated value and computes the likelihood of the alternative models. Through theoretical derivation, we then have the estimated probability of a case read, p, at a given read index, to have a posterior density given by Beta Mixture. We then compute the quantiles of this distribution using a safe version of Newton-Raphson implemented in C as the CI at this read.

Value
- CIRes: A matrix containing the location and its CI of p, each column is a location, or a stretch of location if stepSize>1
- wkRes: The likelihood ratio for alternatives around each estimated change point
- mixStruct: The Beta mixture components for each unique location, containing a collection of two shape parameters and the weight wk
- timeCIRes: A list containing the result of the timing of this algorithm

Author(s)
Jeremy J. Shen

See Also
- ScanBIC

Description
Combine the case and control reads; finds the unique read positions and count the number of case and control reads.

Usage
```r
CombineCaseControlC(cases, controls)
```

Arguments
- cases: A vector of numeric read positions from case sample
- controls: A vector of numeric read positions from control sample

Details
A few C functions are used for efficient implementation
CombineReadsAcrossRuns

Value

- combX: Number of total reads at read position
- combZ: Number of case reads at read position
- combL: Vector of unique read positions

Author(s)

Jeremy J. Shen

See Also

ScanCBS

---

CombineReadsAcrossRuns

*Combine multiple read lists*

Description

Combines multiple lists in the same format of the same sample into one list of the said format.

Usage

```
CombineReadsAcrossRuns(seqs)
```

Arguments

- seqs: A list of lists, each containing equal number of numeric vectors that can be concatenated together. Both number of lists and number of variables can be arbitrary.

Value

Returns a list of the same format as the input lists

Author(s)

Jeremy J. Shen

See Also

ScanCBS
getAutoGridSize

Get Automatic Grid Sizes

Description

This produces a default set of grid sizes to be used in Interactive Grid Scan.

Usage

getAutoGridSize(nL)

Arguments

nL  Number of unique read positions

Details

The default grid sizes are powers of 10.

Value

numeric vector of grid sizes

Author(s)

Jeremy J. Shen

See Also

ScanCBS

Examples

## Should produce a vector of power of ten up to 10000
getAutoGridSize(2*10^5)
getCountsInWindow

Get number of reads in fixed-width window

Description
Computes the number of reads for each fixed-width window between two limits

Usage
getCountsInWindow(events, startE, endE, windowSize = 10000, sorted = FALSE)

Arguments
- events: A vector of the read positions
- startE: Left limit
- endE: Right Limit
- windowSize: Size of the window
- sorted: Whether events is sorted, default F

Details
Uses hist() function

Value
A vector of counts for each window

Author(s)
Jeremy J. Shen

See Also
ScanCBS

Examples
getCountsInWindow(sample(1:10000, 3000, replace=TRUE), 0, 10000, 100, FALSE)
Simulate a homogeneous Poisson Process

Description
Simulation of a homogeneous poisson process using poisson and uniform distribution

Usage
hppSimulate(lambda, maxVal)

Arguments
lambda
  The rate of the poisson process
maxVal
  The maximum length of the process to be observed

Details
This is a very simple simulation function meant to be used in the NHPP generation.

Value
Returns a vector of point events generated for this process

Author(s)
Jeremy J. Shen

See Also
nhppSimulate

Meta File for Simulated Datasets

Description
This data set contains the links and description for the normal and tumor simulated data sets

Usage
JSSim_Meta

Format
A matrix with 3 columns and 4 rows, and a header line
### JSSim_NormalSim1

**Simulated normal sample dataset 1**

**Description**
This data set contains simulated reads of a truncated chromosome from a normal sample.

**Usage**

JSSim_NormalSim1

**Format**

A matrix with 3 columns and 15193 rows

### JSSim_NormalSim2

**Simulated normal sample dataset 2**

**Description**
This data set contains a second set of simulated reads of a truncated chromosome from a normal sample.

**Usage**

JSSim_NormalSim2

**Format**

A matrix with 3 columns and 15206 rows

### JSSim_SpikeMat

**True Signal Spike for the Simulated Dataset**

**Description**
This data set gives the true signal spike location and strength for the simulated datasets.

**Usage**

JSSim_SpikeMat

**Format**

A matrix with 5 columns and rows
**nhppRateEstimate**

<table>
<thead>
<tr>
<th>nhppRateEstimate</th>
<th>Estimate the rate of non-homogeneous PP with data</th>
</tr>
</thead>
</table>

**Description**

Given a vector of point events, give a rough estimate of the rate of underlying non-homogeneous Poisson process by window and smoothing

**Usage**

nhppRateEstimate(controls, length.out = floor(length(controls)/20), lowessF = 0.1)
**Arguments**

- **controls**
  A vector of point locations (read positions) of a control sample for which the rate is wanted

- **length.out**
  The number of windows to be used for the rate estimate vector; default to be number of observations/100

- **lowessF**
  Smoothing factor for the lowess smoothing of the windowed rates, describes the proportion of windows around a particular window that has influence on its smoothed rate estimate

**Details**

This is used to give a realistic estimate of the rate nhpp of control samples

**Value**

Returns a vector of length `length.out` that contains the smoothed rate estimate of each window

**Author(s)**

Jeremy J. Shen

**See Also**

- `nhppSimulate`

---

**nhppSimConstWindowAnalysis**

*Analyze the performance on simulation with constant signal length in each set*

**Description**

Takes the dataset and metafile output of `nhppSimConstWindowGen` and of SegSeq, then evaluates the performance in change-point precision and recall. The dataset must be generated in such format for this function to work.

**Usage**

```r
nhppSimConstWindowAnalysis(filePrefix, chromosomeN, distMetric=c(20,50,100,150,200,300,500,1000), cp...
```
Arguments

filePrefix  The first part of the filename for data and metafile generated by \texttt{nhppSimConstWindowGen}

chromosomeN  The number indicating the chromosome number the dataset emulates

distMetric  A set of criterions of determining change points called are true. A call is deemed true if an actual signal change points within x number of reads is matched to it, after a minimum-cost bipartite matching. Larger value is a looser criterion.

cptLen  The second part of the filename for data and metafile generated by \texttt{nhppSimConstWindowGen}, indicating the length of the true signal. Constant width of the signal (CN gain or loss) region to simulate, can be a vector of different values for which to test

nPair  A part of the filename for data and metafile generated by \texttt{nhppSimConstWindowGen}, indicating the number of normal/tumor pair. Number of tumor samples to generate for each choice of the width of the signal; number of normal samples to generate

nRepeat  A part of the filename for data and metafile generated by \texttt{nhppSimConstWindowGen}, Number of times to repeat the simulation data generation

statistic  The type of statistic to use for the analysis

grid.size  Argument to \texttt{ScanCBS}

takeN  Argument to \texttt{ScanCBS}

maxNCut  Argument to \texttt{ScanCBS}

minStat  Argument to \texttt{ScanCBS}

verbose  If True, will print run information as the algorithm proceeds

timing  Performs timing of the \texttt{ScanCBS} algorithm

hasRun  If True, will read the output file of \texttt{ScanCBS} instead of run it on these datasets again. Only use when the same call to \texttt{ScanCBS} has been used before in this function call.

width  Width of the graph output file

height  Height of the graph output file

Details

This function is used in conjunction with \texttt{nhppSimConstWindowGen}. It reads in the data and metafile output of the said function, and compares the performance of our algorithm with SegSeq. It is important that SegSeq has been used on the simulation datasets generated before using this.

Value

\texttt{simCBS}  Result of \texttt{ScanCBS} output structure

\texttt{CBSMatchDist}  The distance among reads after minimum-cost bipartite graph matching for our algorithm

\texttt{SegMatchDist}  The distance among reads after minimum-cost bipartite graph matching for SegSeq

\texttt{CBSRecall, SegRecall}  The recall rates of two algorithms
nhppSimConstWindowGen  

CBSPrecision, SegPrecision
  The precision rates of two algorithms
CBSFMeasure, SegFMeasure
  The F-measure of two algorithms
trueTauMeanSigLen
  The mean distance between true signal boundaries
nTrueTau
  The number of true change points
nCBSCall, nSegCall
  Number of change points called by the two algorithms
CBSTime
  Mean computational time of ScanCBS for each signal length

Author(s)
  Jeremy J. Shen

See Also
  nhppSimConstWindowGen

nhppSimConstWindowGen  Simulate a Non-Homogeneous PP with constant window spike  

Description
  Simulate non-homogeneous Poisson processes with a number of constant-widths windows of signal spike, and output the data and meta file

Usage
  nhppSimConstWindowGen(controlRates, filename, chromosomeN, nSpike=25, cptLen=c(3,5,8,12,20,30,50,75),
         npair=1, nRepeat=1, mingain=1, maxgain=100, minloss=0, maxloss=100, npair=100)

Arguments
  controlRates       The estimated rate of nhpp for the control
  filename           The prefix of all the output files from this simulation
  chromosomeN       The chromosome number. Should be the number from which the samples are emulated
  nSpike             Number of signal spikes
  cptLen              Constant width of the signal (CN gain or loss) region to simulate, can be a vector of different values for which to test
  npair               Number of tumor samples to generate for each choice of the width of the signal; number of normal samples to generate
  nRepeat            Number of times to repeat the simulation data generation
  minGain             Minimal signal gain
NhppSimulate

maxGain  Maximal signal gain
minLoss  Minimal signal loss
maxLoss  Maximal signal loss
pgain    Proportion of the signal regions that are CN gain

Details
This function is used in conjunction with a modified, windowed rate vector to simulate non-homogeneous Poisson processes with a number of constant-widths windows of signal spike. One should use the nhppRateEstimate function to estimate the rate of a control sample one wishes to mimic. This function randomly choose windows of a specified constant width, and spike in signals (change points) which can be either gain or loss of copy numbers.

Value
No return value. Generates a number of .txt files, one for each normal/tumor sample as raw data, one input meta file and a file with the true change points for each choice of cptLen.

Author(s)
Jeremy J. Shen

See Also
nhppSimulate

nhppSimulate  Simulate a non-homogeneous Poisson Process

Description
This function simulates an NHPP by blocked thinning

Usage
nhppSimulate(smoothRates)

Arguments
smoothRates  A list containing x and y, which are the mid-points of the window and the smoothed number of events in this window

Details
The list component y of the argument represents the smoothed number of events in the window, namely, they represent the window rate
Value

Returns a vector of events of a realization of the NHPP

Author(s)

Jeremy J. Shen

See Also

nhppRateEstimate, nhppSpike

NHPP Spike

Spike rates of NHPP

Description

Randomly spike the smoothed control rate of an NHPP according to the parameters.

Usage

nhppSpike(smoothRates, nSpike = 25, cptLenR = 4, cptLenMean = 10, minGain = 1.5, maxGain = 10, minLoss = 1, maxLoss = 1, pgain = 0.61)

Arguments

smoothRates The smoothed rate estimate of the control process
nSpike Number of signal spikes
cptLenR Parameter for signal width (parameter R of negative binomial)
cptLenMean Parameter for signal width (mean width)
minGain Minimal Gain relative CN
maxGain Maximal Gain relative CN
minLoss Minimal Loss relative CN
maxLoss Maximal Loss relative CN
pgain Proportion of signal regions that are CN gain.

details

The signal width is randomly generated by negative binomial distribution with the two parameters given. The signal strength are uniformly drawn between the two limits.

Value

spikeMat A matrix containing the actual signal spike information
caseRates The rate of the case PP to be simulated after signal spike
Author(s)
Jeremy J. Shen

See Also
nhppSimulate

---

**nhppSpikeConstWindow**

*Spike NHPP rate with constant window width*

**Description**
The signal strength are uniformly drawn between the two limits.

**Usage**
nhppSpikeConstWindow(smoothRates, nSpike = 25, cptLen = 5, minGain = 1.5, maxGain = 10, minLoss = 0.01)

**Arguments**
- smoothRates: The smoothed rate estimate of the control process
- nSpike: Number of signal spikes
- cptLen: Window width of each signal region
- minGain: Minimal Gain relative CN
- maxGain: Maximal Gain relative CN
- minLoss: Minimal Loss relative CN
- maxLoss: Maximal Loss relative CN
- pGain: Proportion of signal regions that are CN gain

**Details**
The signal strength are uniformly drawn between the two limits.

**Value**
- spikeMat: A matrix containing the actual signal spike information
- caseRates: The rate of the case PP to be simulated after signal spike

**Author(s)**
Jeremy J. Shen

See Also
nhppSimulate
readInput

*Manage reading and merging of raw datasets. Main file input*

**Description**

This is used to control the read of a meta file containing names of data files, merge, and give usable output for the main program.

**Usage**

```
readInput(inputFilename, formatName="Chiang", sep = "\t")
```

**Arguments**

- `inputFilename`: The name of file, containing relevant information of all input files.
- `formatName`: The format in which the data files are written in. We use the simple ‘Chiang’ as default format of input.
- `sep`:Delimiter of the meta input file, default is tab-delimited.

**Details**

The meta input file should be organized in a table format with 2 columns, one of which is 'file' and the other is 'type', indicating the data file names and whether the data is from normal or tumor. We recommend using the ‘Chiang’ format, as used by the datasets of Chiang (2009). This format requires minimal memory and contains all relevant information for this program. It is a table with two columns, first being the chromosome of the mapped read, and the second being the position of the read in the chromosome. One line for each observation.

**Value**

- `normalSeq`: A list containing the combined normal/control reads
- `tumorSeq`: A list containing the combined case/tumor reads

**Author(s)**

Jeremy J. Shen

**See Also**

`readListInputFile, readSeq`
Examples

# This shows the format of the meta file
data(JSSim_Meta)
print(JSSim_Meta)

# This shows the recommended format, the Chiang data format
data(JSSim_NormalSim1)
print(head(JSSim_NormalSim1))

readListInputFile  
\textit{Read meta file containing list of raw data files}

Description

Reads a meta file that contains the file names and type of the data files. See details for the format.

Usage

readListInputFile(inputFilename, sep = "\t")

Arguments

inputFilename  The name of file, containing relevant information of all input files
sep  Delimiter of the meta input file, default is tab-delimited

Details

The meta input file should be organized in a table format with 2 columns, one of which is 'file' and the other is 'type', indicating the data file names and whether the data is from 'normal' or 'tumor'.

Value

normalFiles  A character vector containing the names of files with the normal reads
tumorFiles  A character vector containing the names of files with the tumor reads

Author(s)

Jeremy J. Shen

See Also

\textit{readInput, readSeq}

Examples

# This shows the format of the meta file
data(JSSim_Meta)
print(JSSim_Meta)
**Description**

This is a wrapper function. It calls one of the subroutines to reads in a datafile, depending on the format.

**Usage**

```r
readSeq(filename, formatName)
```

**Arguments**

- `filename`: The file name of the data file to be read
- `formatName`: The format the file is in. Can be either 'Chiang' or 'ELANDPaired'. We recommend using Chiang since this is the minimal required format.

**Details**

We recommend using the 'Chiang' format, as used by the datasets of Chiang (2009). This format requires minimal memory and contains all relevant information for this program. It is a table with two columns, first being the chromosome of the mapped read, and the second being the position of the read in the chromosome. One line for each observation. If one has paired read, please use only one of the reads and the mapped location should be the 5'-end.

**Value**

- `seqF`: Read position for each read
- `seqChr`: Chromosome of each mapped read

**Author(s)**

Jeremy J. Shen

**References**

Chiang et al., Nature Methods, 2009, Vol.6 No.1

**See Also**

`readSeq`, `readSeqChiang`, `readSeqELANDPaired`

**Examples**

```r
# This shows the recommended format, the Chiang data format
data(JSSim_NormalSim1)
print(head(JSSim_NormalSim1))
```
readSeqChiang  

*Read data formatted as in Chiang (2009)*

---

**Description**

Read data formatted as in Chiang (2009), which we recommend using.

**Usage**

`readSeqChiang(filename)`

**Arguments**

- `filename` The file name of the data set

**Details**

This format requires minimal memory and contains all relevant information for this program. It is a table with two columns, first being the chromosome of the mapped read, and the second being the position of the read in the chromosome. One line for each observation. In case of paired read, we only use the front read (whichever has a smaller position label) and ask that you use only that for input.

**Value**

- `seqF` Read position for each read
- `seqChr` Chromosome of each mapped read

**Author(s)**

Jeremy J. Shen

**References**

Chiang et al., Nature Methods, 2009, Vol.6 No.1

**See Also**

`readSeq`, `readSeqChiang`, `readSeqELANDPaired`

**Examples**

```r
# This shows the format of this type of data
data(JSSim_NormalSim1)
print(head(JSSim_NormalSim1))
```
**readSeqELANDPaired**

Read raw data formatted as in paired ELAND output

**Description**

Read datasets with paired-end format, possible output format of ELAND

**Usage**

`readSeqELANDPaired(filename)`

**Arguments**

filename  
The file name of the data set

**Details**

This format has two reads per line, each looking like `NACGATGAAACCCCGTCTCTACTAAC-CATACAAAAA hs_ref_chr17.fa 12091150 R TGTCGCCCAGGCTGCAATGCAGTGGCGCGATCTCGG hs_ref_chr17.fa 12091018 F'`. There are 8 columns, 4 for each of the paired read. The first is the actual read sequence, which we discard; the second is the chromosome of the mapped read; the third is the read position; and the last is indicating whether it is a front- or rear-end read. We only use the reads with the same mapped chromosome and only the front read. This contains more information than needed; the Chiang format is preferred.

**Value**

- seqF: Read position for each read
- seqChr: Chromosome of each mapped read

**Author(s)**

Jeremy J. Shen

**See Also**

`readSeq`, `readSeqChiang`
relCNComp

*Compute the Relative Copy Number*

**Description**
This computes the relative copy number by each of the segment called.

**Usage**

\[
\text{relCNComp}(\text{combx}, \text{combz}, \text{tauHatInd}, p, \text{alpha})
\]

**Arguments**
- **combx**: The number of reads at each unique read position
- **combz**: The number of case/tumor reads at each unique read position
- **tauHatInd**: The index of change points called
- **p**: The overall proportion of case reads
- **alpha**: Significance level for testing whether each segment is a gain (relative CN > 1) or loss (relative CN < 1). The method internally corrects for multiple testing.

**Details**
The relative CN is defined as the number of case reads divided by the number of control reads in a window, adjusted for overall proportion of case reads (divided by the overall relative CN).

**Value**
Returns a vector of relative CN for each of the segment between two change points.

**Author(s)**
Jeremy J. Shen

ScanBIC

*Compute the modified BIC for change-point models*

**Description**
This computes mBIC for the current change point model. We then use this to determine the appropriate model complexity.

**Usage**

\[
\text{ScanBIC}(\text{combx}, \text{combz}, \text{tauHat}, \text{lik0}, \text{nTotal})
\]
**Arguments**

- `combX`: The number of reads at each unique read position
- `combZ`: The number of case/tumor reads at each unique read position
- `tauHat`: The change points called
- `lik0`: The null likelihood. Computed in the main routine.
- `nTotal`: The total number of reads

**Details**

This is meant to be called as a subroutine of `ScanCBS`

**Value**

Returns a numerical value of mBIC for the current model

**Author(s)**

Jeremy J. Shen

**See Also**

`ScanCBS`

**Description**

This is the main algorithm. It iteratively scans for window of arbitrary size where the case and control read depths are different. It continues until a stopping criterion based on mBIC, maximum number of cut, and the statistic at the current segment.

**Usage**

`ScanCBS(cases, controls, statistic = "binomial", grid.size = "auto", takeN = 5, maxNCut = 100, minStat = 1.5, verbose = FALSE)`

**Arguments**

- `cases`: A numeric vector of the case/tumor reads
- `controls`: A numeric vector of the control/normal reads
- `statistic`: The statistic to be used. Can be 'binomial', 'rabinowitz' or 'normal'.
- `grid.size`: The set of grid sizes for the iterative search. An automatic default can be computed.
- `takeN`: The number of candidate change points to be added to a temporary set at each grid size
maxNCut  The maximum number of segmentation steps to perform
minStat  The minimum statistic value required to continue the segmentation. Default 0 as this criterion being ignored.
alpha   Significance level for testing whether each segment is a gain (relative CN > 1) or loss (relative CN < 1). The method internally corrects for multiple testing.
verbose If TRUE, then will print much information on each segmentation. For diagnostics only.
timing  If TRUE, perform a timing of this algorithm, include in the output data file.

Details
This algorithm is an use of the Circular Binary Segmentation method. It continues to segment the reads and consider the resulting child regions for further segmentation. It keeps track of the most promising cut in each children, and only the child region with the most significant segmentation is further cut, yielding more children. This is repeated until stopping criteria are met. The three types of statistics are by the use of exact binomial likelihood ("binomial"), score statistic ("rabinowitz") or using normal approximation to the binomial ("normal").

Value
tauHat  The change points called
statHat A matrix containing the statistic and its segmentation for the model called, in the order of the segmentation. The columns are break points in genomic scale (1,2), read index scale (3,4), value of test statistic (5), the parent segment in genomic scale (6,7), and mBIC of the model (8).
relCN   The relative CN computed for each segment between change points
relGainLoss Test result of whether each segment is a gain, loss, or normal
timingRes A list containing the result of the timing of this algorithm

Author(s)
Jeremy J. Shen

References
D. Rabinowitz, IMS Lecture Notes - Monograph Series, Vol. 23, 1994

See Also
ScanIterateGrid, ScanBIC, relCNComp, getAutoGridSize
ScanCBSPlot  

Main Plotting of the scan statistic segmentation

Description

This is an overall plotting function to display the segmentation for a chromosome.

Usage

ScanCBSPlot(cases, controls, CBSObj, filename, mainTitle, CIObj=NULL, length.out=10000, localWindow=0, localSeparatePlot=TRUE, smoothF=0.5, xlabScale=1, width=10, height=5)

Arguments

cases The case read positions (should be restricted to a chromosome)
controls The control read positions (should be restricted to a chromosome)
CBSObj The output object of the ScanCBS function
filename The output file names of the plot
mainTitle The title of the plot
CIObj Optional; the Bayesian CI computed by BayesCptCI function
length.out The number of windows to use for the display of smoothed rate estimates
localWindow The number of genome locations to show around each of the called change points
localSeparatePlot Whether to show the local behavior of each change point in a separate PDF file. Default to TRUE. The output file are the given filename attached with the index and actual location of the change point.
smoothF The lowess smoothing factor. The proportion of windows around the current window that affects its smoothed rate estimate
xlabScale The scaling factor of the read positions, often in 10^6, or Mb
width The width of the output graph in inches
height The height of the output graph in inches

Details

This function produces three sub-graphs, showing the segmentation calls, the smoothed rate estimate, and the inferred relative copy number. It is crucial that one separates the plot for each chromosome. It also makes a zoom-in plot for a region around each of the called change points.

Value

No return object

Author(s)

Jeremy J. Shen
See Also

ScanCBS, ScanCBSSimPlot, relCNComp

ScanCBSSimPlot  
Plotting for CBS results of Simulated Data

Description

This is an overall plotting function to display the segmentation for a chromosome, for simulation data.

Usage

ScanCBSSimPlot(cases, controls, CBSObj, trueTau, SpikeMat, filename, maintitle, CIObj=NULL, lengthNout=1, localwindow=PNUJ1P, localseparateplot=true, smoothf=PNPRUL, xlabscale=1PV, width=1RL, height=1XI)

Arguments

cases  
The case read positions (should be restricted to a chromosome)
controls  
The control read positions (should be restricted to a chromosome)
CBSObj  
The output object of the ScanCBS function
trueTau  
The true location of the change points in simulation
SpikeMat  
The matrix of signal spikes as generated by the relevant simulation functions
filename  
The output file names of the plot
maintitle  
The title of the plot
CIObj  
Optional; the Bayesian CI computed by bayescptci function
lengthNout  
The number of windows to use for the display of smoothed rate estimates
localwindow  
The number of genome locations to show around each of the called change points
localseparateplot  
Whether to show the local behavior of each change point in a seperate PDF file. Default to TRUE. The output file are the given filename attached with the index and actual location of the change point.
smoothf  
The lowess smoothing factor. The proportion of windows around the current window that affects its smoothed rate estimate
xlabscale  
The scaling factor of the read positions, often in 10^6, or Mb
width  
The width of the output graph in inches
height  
The height of the output graph in inches

Details

This is similar to ScanCBSPlot. This function produces three sub-graphs, showing the segmentation calls, the smoothed rate estimate, and the inferred relative copy number. It is crucial that one seperates the plot for each chromosome. This also has an option of showing each change point details in seperate graphs.
ScanIterateGrid

Value

No return object

Author(s)

Jeremy J. Shen

See Also

ScanCBS, ScanCBSPlot, relCNComp

ScanIterateGrid  Main Scan with Iterative Grid Search

Description

This is a computational speed-up to prevent a quadratic order computation.

Usage

ScanIterateGrid(combX, combZ, combL, statistic, grid.size, nGridSize, timeIGSBreakDown, takeN, verbose)

Arguments

<table>
<thead>
<tr>
<th>Argument</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>combX</td>
<td>The number of reads at the unique read positions</td>
</tr>
<tr>
<td>combZ</td>
<td>The number of case reads at the unique read positions</td>
</tr>
<tr>
<td>combL</td>
<td>The set of the labels for the unique read positions</td>
</tr>
<tr>
<td>statistic</td>
<td>The type of statistic to be used. Can be 'binomial', 'rabinowitz', or 'normal'</td>
</tr>
<tr>
<td>grid.size</td>
<td>The set of grid sizes for the iterative search. An automatic default can be given</td>
</tr>
<tr>
<td>nGridSize</td>
<td>The number of grid sizes</td>
</tr>
<tr>
<td>timeIGSBreakDown</td>
<td>Cumulative timing of IGS, in a broken down fashion</td>
</tr>
<tr>
<td>takeN</td>
<td>The number of candidate change points to be added to a temporary set at each grid size</td>
</tr>
<tr>
<td>verbose</td>
<td>If TRUE, then will print much information on each segmentation. For diagnostics only.</td>
</tr>
<tr>
<td>timing</td>
<td>If TRUE, perform a timing of this algorithm, include in the output data file.</td>
</tr>
</tbody>
</table>

Details

This algorithm is a computational speed-up tool. It computes the statistic on coarse grids, and refine to finer grids. Also, at each refinement, it computes all new smaller windows on the finer grid that would not have been captured by the coarse grid. Hence it has a New Scan step and a Refine Scan step, both implemented in C for speed. The three types of statistics are by the use of exact binomial likelihood ('binomial'), score statistic ('rabinowitz') or using normal approximation to the binomial ('normal').
Value

cptsRet The current set of change points called after the IGS scan of the current region
timeIGSBreakDown A break-down of the time used at the stages of the IGS

Author(s)

Jeremy J. Shen

See Also

ScanCBS, ScanStatNewComp, ScanStatRefineComp

ScanStatNewComp Main new window scan statistics computation

Description

This is a wrapper function to call the C routines for the scan statistic new candidate segmentation computing from the IGS

Usage

ScanStatNewComp(combZCumSum, combXCumSum, combZPoint, combXPoint, p, nTotal, grid.cur, max.win, statistic)

Arguments

combZCumSum A cumulative sum of the number of case reads
combXCumSum A cumulative sum of the number of reads
combZPoint The number of case reads at the grid points
combXPoint The number of reads at the grid points
p The proportion of case reads in the current region
nTotal The total number of reads in the current region
grid.cur The current grid to be computed on
max.win The maximum inter-window to be considered for new scan
statistic The type of statistic. Can be 'binomial','rabinowitz' or 'normal'

Details

The computations are done in C for speed. The three types of statistics are by the use of exact binomial likelihood ('binomial'), score statistic ('rabinowitz') or using normal approximation to the binomial ('normal').
**ScanStatRefineComp**

**Value**

Returns a matrix containing the candidate change points from the new scan

**Author(s)**

Jeremy J. Shen

**See Also**

`ScanCBS, ScanIterateGrid`

**Description**

This is a wrapper function to call the C routines for the scan statistic to refine current candidate segmentations computing from the IGS

**Usage**

`ScanStatRefineComp(combZCumSum, combXCumSum, combZPoint, combXPoint, p, nTotal, grid.cur, grid.LR, maxNwin, statistic)`

**Arguments**

- `combZCumSum`: A cumulative sum of the number of case reads
- `combXCumSum`: A cumulative sum of the number of reads
- `combZPoint`: The number of case reads at the grid points
- `combXPoint`: The number of reads at the grid points
- `p`: The proportion of case reads in the current region
- `nTotal`: The total number of reads in the current region
- `grid.cur`: The current grid to be computed on
- `grid.LR`: The left and right limits of the existing candidate segmentations that will be refined, indexed by the current grid
- `maxNwin`: The maximum inter-window to be considered for new scan
- `statistic`: The type of statistic. Can be 'binomial', 'rabinowitz' or 'normal'.

**Details**

The computations are done in C for speed. The three types of statistics are by the use of exact binomial likelihood ('binomial'), score statistic ('rabinowitz') or using normal approximation to the binomial ('normal').
Value
Returns a matrix containing the refined candidate change points

Author(s)
Jeremy J. Shen

See Also
scanCBS, ScanIterateGrid

---

**SegSeqResProcess**  \hspace{1cm} **Read and Process result of SegSeq**

**Description**
Read the segmentation results of SegSeq and returns the change points called

**Usage**

\[ \text{SegSeqResProcess(filename)} \]

**Arguments**

- **filename**  \hspace{0.5cm} The filename of the SegSeq output file to be processed

**Details**
This function is used to read in the SegSeq results and use for performance evaluation and comparison

**Value**
Return a list the length of unique chromosomes in the result file. For each entry, the label is the chromosome label; and there is a vector of the change point locations called by SegSeq

**Author(s)**
Jeremy J. Shen

**See Also**

nhppSimulate
Index

*Topic datasets
  JSSim_Meta, 8
  JSSim_NormalSim1, 9
  JSSim_NormalSim2, 9
  JSSim_SpikeMat, 9
  JSSim_TumorSim1, 10
  JSSim_TumorSim2, 10

*Topic package
  seqCBS-package, 2

BayesCptCI, 3

CombineCaseControlC, 4
CombineReadsAcrossRuns, 5

getAutoGridSize, 6, 24
getCountsInWindow, 7

hppSimulate, 8
  JSSim_Meta, 8
  JSSim_NormalSim1, 9
  JSSim_NormalSim2, 9
  JSSim_SpikeMat, 9
  JSSim_TumorSim1, 10
  JSSim_TumorSim2, 10

nhppRateEstimate, 10, 14, 15
nhppSimConstWindowAnalysis, 11
nhppSimConstWindowGen, 11–13, 13
nhppSimulate, 8, 11, 14, 14, 16, 30
nhppSpike, 15, 15
nhppSpikeConstWindow, 16

readInput, 17, 18
readListInputFile, 17, 18
readSeq, 17–19, 19, 20, 21
readSeqChiang, 19, 20, 20, 21
readSeqELANDPaired, 19, 20, 21
relCNCComp, 22, 24, 26, 27

ScanBIC, 4, 22, 24
ScanCBS, 5–7, 12, 13, 23, 23, 25–30
ScanCBSPlot, 25, 26, 27
ScanCBSSimPlot, 26, 26
ScanIterateGrid, 24, 27, 29, 30
ScanStatNewComp, 28, 28
ScanStatRefineComp, 28, 29
SegSegResProcess, 30
seqCBS (seqCBS-package), 2
seqCBS-package, 2