Package ‘modSaRa’

November 30, 2016

Type Package

Title modSaRa: a computationally efficient R package for CNV identification

Version 1.0

Depends R (>= 2.10), stats, utils

Imports Rcpp (>= 0.12.6)

Date 2016-03-12

Author Feifei Xiao, Yue Niu, Ning Hao, Yanxun Xu, Zhilin Jin and Heping Zhang

References

1 Xiao F, Min X and Zhang H. (2015) Modified screening and ranking algorithm for copy number variation detection. Bioinformatics. 31(9):1341-8.

2 Hao N, Niu Y and Zhang. (2013) Multiple change-point detection via a screening and ranking algorithm. Statistica Sinica 23.

3 Niu Y and Zhang H. (2012) The screening and ranking algorithm to detect DNA copy number variations. Ann. Appl. Stat. 6,1306-1326.

Maintainer Feifei Xiao <xiaof@mailbox.sc.edu>

Description modSaRa: a computationally efficient R package for CNV identification

SeeAlso http://c2s2.yale.edu/software/modSaRa/

License GPL(>=2)

LinkingTo Rcpp, RcppArmadillo

RoxygenNote 5.0.1

R topics documented:

`CNVcluster` *CNVcluster*

Description

This function uses different priors settings to achieve convergence of the Expectation-Maximization algorithm, and then determine the best clustering results by applying the modified BIC.

Usage

```
CNVcluster(Y, cp, L)
```

Arguments

`Y`	the numeric vector of the intensities of markers
`cp`	the numeric vector of the position index for the identified change-points
`L`	repeat times in the EM algorithm, defaults to 100

Value

The return is the clustered CNV segments by presenting the start position and end position, length of the CNV and the copy number states (duplication or deletion). It also returns a vector of final candidates of change-points.

`newcp`	the final list of change-points
`h`	the bandwidth used for the identification of change-points
`cnv.state`	copy number state for each CNV
`cnv.start`	start position of each CNV
`cnv.end`	end position of each CNV

See Also

gausianMixure for clustering of CNVs using Expectation-Maximization algorithm.

CNVout *CNVout*

Description

This function annotates the identified CNV using the reference map file and output the annotation of all identified CNVs. Each line of the output describes one CNV in nine columns: individual ID; chromosome ID; CNV start marker identifier; CNV start location (in base pair units); CNV end marker identifier; CNV end location (in base pair units); length of CNV (in base pair units); length of CNV(number of markers); copy number states (duplication or deletion).

Usage

```
CNVout(lrr, map, h1 = 5, h2 = 10, h3 = 15, alpha = 0.01, L = 100,
  outname)
```

Arguments

`lrr`	the matrix of the log R ratio intensities. Each column describes a single sample or sequence and each row describes a single marker
`map`	Each line of the map file describes a single marker and must contain exactly 3 columns: chromosome ID; rs# or marker identifier; position (in bp units)
`h1`	the bandwidth 1 for the screening procedure, defaults to 5
`h2`	the bandwidth 2 for the screening procedure, defaults to 10
`h3`	the bandwidth 3 for the screening procedure, defaults to 15
`alpha`	the significance levels for the test to accept change-points
`L`	number of iterations in the EM algorithm for CNV clustering
`outname`	name for the output file

Value

This function generates a text file describing all detected CNVs. In addition, it also returns a list of detected change-points for all samples.

`cp`	a list of position index for the final change-points identified by modSaRa

See Also

modifiedSaRa for processing the modified SaRa method

Examples

```
# Input the example data of SNP genotyping data from Affymatrix Human SNP Array 6.0 platform.
# The map file displays annotation of the markers including the chromosome and location
# information of each SNP or CNV marker.
data(example.data.lrr)
data(example.data.map)
# Use log R ratio information of ten samples to detect CNVs
cnv.out <- CNVout(lrr=example.data.lrr[,1:10],map=example.data.map, outname="out")
# The following file will be generated: "out.csv<e2><80><9d>
# This file contains CNV output for each individual.
# Each line represents one CNV detected from one sample or sequence.
# For each line, the individual ID, start position, end position, length and state
# (duplication or deletion) of the CNV will be shown.
out.cp <- cnv.out$cp
# This returns a list of vectors containing detected change-points by modSaRa for each
# sample in the marker name.
```

`estimateSigma` *Estimate the standard deviation of the intensities between two adjacent change-points*

Description

This function estimates the standard deviation between two adjacent change-points using a local smoother.

Usage

```
estimateSigma(Y, h = 10)
```

Arguments

`Y`	the numeric vector of the intensities of markers
`h`	the bandwidth of the local smoother

Value

This function estimates the standard deviation of the intensities between two adjacent change points

---

`example.data.lrr` *example.data.lrr*

Description

An example data to demonstrate modSaRa processing multiple sequences

Usage

```
data(example.data.lrr)
```

Format

A data frame with 10 individuals on 50000 markers

Details

This example data is normalized measure of signal intensities for 10 sequences. Each column represents data for one individuals and each row represents data for one marker.

Examples

```
# input the data
data(example.data.lrr)
data <- example.data.lrr
```

`example.data.map` *example.data.map*

Description

An example map file to demonstrate modSaRa

Usage

```
data(example.data.map)
```

Format

A data frame with 50000 markers on the following three variables: Name as SNP identifier; Chr as chromosome ID; Position as the chromosomal coordinates.

Examples

```
# input the data
data(example.data.map)
# get general information about the three variables
str(example.data.map)
```

`fInverse` *Get the inverse cumulative distribution function of local min p-values*

Description

This function approximates the inverse cumulative distribution function (CDF) of the local min p-values via simulation.

Usage

```
fInverse(n = 10000, h = 10, hh = 2 * h, precise = 10000, simT = 100)
```

Arguments

`n`	the length of the data to simulate
`h`	the bandwidth for the screening procedure. Defaults to 10
`hh`	the bandwidth for the local minimum procedure
`precise`	the precision of the approximation (the number of quantiles to use)
`simT`	number of simulations

Value

the quantiles of the approximated inverse CDF

`gausianMixure` *Clustering of CNVs using Expectation-Maximization algorithm*

Description

This function clusters the identified change-points to make final CNV calling.The potential CNV segments between two neighbor candidate change-points are assigned to different copy number states according to the average intensities in the segment intervals. We use three clusters including duplication, normal state and deletion. Each cluster is presented by Gaussian distribution with unknown mean and variance. Expectation-Maximization (EM) algorithm is applied for the mixture of Gaussians to assign each segment to the most probable cluster/state. Two physically linked candidate CNV segments in the same group are merged to one unique CNV segment.

Usage

```
gausianMixure(x, cp, priors, L, st)
```

Arguments

`x`	the vector of the intensities of markers
`cp`	the vector of the marker index of the identified change-points
`priors`	given initial parameters for the EM algorithm
`L`	repeat times in the EM algorithm. Defaults to 100
`st`	number of assumed states in the EM algorithm

Value

The return is the clustered CNV segments by presenting the start position and end position using SNP or CNV marker index, and the copy number states. It also returns a vector of final candidates of change-points.

`p.final`	probability of falling into each state for each CNV segment after convergence
`mu.final`	segment means of each state after convergence
`cp.final`	list of change-points after EM algorithm
`index.final`	the bandwidth of change-points
`state.new`	assigned copy number state for each CNV

`getOneBIC` *The modified Bayesian-type Information Criterion step to remove false positives in change-points*

Description

The modified Bayesian-type Information Criterion (BIC) step to remove false positives in change-points

Usage

```
getOneBIC(x, cp, mod = FALSE)
```

Arguments

`x`	the vector of the intensities of markers
`cp`	the vector of the marker index of the identified change-points

Value

a list of change-points after filtering false positives

`localDiagnostic` *Calculate the value for local diagnostic function*

Description

This function calculates local diagnostic function D(x,h) at each point x which depends only on observations in a small neighborhood [x-h,x+h].

Usage

```
localDiagnostic(y, h)
```

Arguments

`y`	the numeric vector of the intensities of markers
`h`	the bandwidth for the screening procedure

Details

Local diagnostic function reflects of position x being or neighborhooding a change-point. A reasonable local diagnostic is

$$D(x) = \frac{\Sigma_{k=1}^{h} y_{x+k} - \Sigma_{k=1}^{h} y_{x+1-k}}{h}$$

which is the difference between averages of h data points on the left side and right side of x. Suppose the errors $\varepsilon_i = 0$ which means $Y = \mu$ is a piecewise constant vector and D(x) is piecewise linear function. Based on this function, we proposed a recursive formula

$$D(x+1)_h = D(x)_h + \frac{Y_{x-h+1} + Y_{x+h+1} - 2Y_{x+1}}{h}$$

Value

This function generates a numeric vector of local diagnostic function values D(x,h) at each point x

localMax *Get the local maximizers of local diagnostic function*

Description

This function finds the local maximizers of local diagnostic function |D(x,h)| and returns the value of position index for all the local maximizers.

Usage

```
localMax(y, span = 5)
```

Arguments

`y`	the list of local diagnostic values within a small neighborhood [x-h, x+h]
`span`	the bandwidth to find local Maximizer of local diagnostic function

Details

Local maximizer is defined as follows. For any number x, the interval (x-h, x+h) is called the h-neighborhood of x. And, x is an h-local maximizer of function f(.) if reaches the maximum at x within the h-neighborhood at x. In other words, f(x)>=f(x') for all x' in (x-h, x+h).

Value

numeric vector of position index for all the local maximizers of local diagnostic function

modifiedSaRa *CNV detection processing multiple sequences using the modified SaRa algorithm*

Description

This function runs the modified SaRa algorithm and cluster the change-points to CNVs processing multiple sequences.

Usage

```
modifiedSaRa(Y, alpha = 0.01, L = 100, h1 = 5, h2 = 10, h3 = 15,
  precise = 10000, simT = 100, sigma = NULL)
```

Arguments

`Y`	the numeric vector of the intensities of markers
`alpha`	the significance levels for the test to accept change-points
`L`	number of iterations in the EM algorithm for CNV clustering
`h1`	the bandwidth 1 for the screening procedure, defaults to 5
`h2`	the bandwidth 2 for the screening procedure, defaults to 10

`h3`	the bandwidth 3 for the screening procedure, defaults to 15
`precise`	the precision of the inverse CDF of local min p-values. This will be used only if FINV is not specified. Defaults to 10000
`simT`	number of simulations in getting the inverse CDF of the local minimum p values
`sigma`	the standard deviation for the intensities between two adjacent change-points, defaults to NULL

Value

This function generates a list of detected change-points and clustered CNVs for all samples.

`newcp`	a list of vectors presenting detected change-points, which is in marker index units. Length of the list is the number of samples or sequences
`h`	a list of vectors presenting the bandwidth used for this detected change-points. Length of the list is the number of samples
`cnv.state`	state of detected CNV segments, duplication or deletion
`cnv.start`	a list of vectors presenting the start position of CNV segments
`cnv.end`	a list of vectors presenting the end position of CNV segments

See Also

multiSaRa for processing the screening and ranking steps for single sequence

---

`multiSaRa` *Screening procedure processing single sequence to find local maximizers of the local dignostic statistic*

Description

This function runs the screening step under multiple bandwidths processing a single sequence.

Usage

```
multiSaRa(Y, h1 = 3 * round(log10(length(Y))), h2 = 2 *
  round(log10(length(Y))), h3 = round(log10(length(Y))), FINV = FINV,
  precise = precise, sigma = sigma)
```

Arguments

`Y`	the vector of the intensities of markers
`h1`	the bandwidth 1 for the screening procedure, defaults to 5
`h2`	the bandwidth 2 for the screening procedure, defaults to 10
`h3`	the bandwidth 3 for the screening procedure, defaults to 15
`FINV`	the inverse CDF of the local minimum p-values, approximated by function fInverse()
`precise`	the precision of the inverse CDF of local min p-values. This will be used only if FINV is not specified. Defaults to 10000
`sigma`	the standard deviation for the intensities between two adjacent change-points, defaults to NULL

Value

The return is a list of index with local minimum p values at each bandwidth.

See Also

SARA for processing the screening and ranking steps using single bandwidth

---

QuanNorm *Quantile normalization of the original intentisites*

Description

This function runs the quantile normalization procedure for each sequence separately as a preprocessing step.

Usage

```
QuanNorm(lrr)
```

Arguments

`lrr`	the matrix of intensities. Each column represents a sequence or sample, each row represents a single marker

Value

This function generates a vector of signal intensities for a single sequence after quantile normalization

Examples

```
# Input the example data of SNP genotyping data from Affymatrix Human SNP Array 6.0 platform
data(example.data.lrr)
lrr.qn <- QuanNorm(example.data.lrr) ## quantile normalization of the intensities
```

SARA *Screening procedure processing single sequence and p value calculation of local maximizers*

Description

This function runs the screening and ranking algorithm under single bandwidth processing a single sequence. Local min p-values are corrected by approximation emprically by the distribution of the local minimizers in a long standard normal sequence.

Usage

```
SARA(Y, h = 10, hh = 2 * h, FINV = FINV, sigma = NULL,
  precise = 10000)
```

Arguments

`Y`	the numeric vector of the intensities of markers
`h`	the bandwidth for the screening procedure, defaults to 10
`hh`	the bandwidth for the local minimum procedure
`FINV`	the inverse CDF of the local min p-values, approximated by function fInverse
`sigma`	the standard deviation for the intensities between two adjacent change points, defaults to NULL
`precise`	the precision of the inverse CDF of local minimum p-values. This will be used only if FINV is not specified. Defaults to 10000

Value

The return is a vector of corrected local p-value minimum and the marker index of these minimum.

`index`	index of markers for the corrected local p-value minimizers
`pV`	corrected local min p-values

See Also

SARAp for processing the screening step using single bandwidth to find local maximizers of the diagnostic statistic

---

`SARAp` *Screening procedure processing single sequence to find local maximizers of the dignostic statistic*

Description

This function runs the screening procedure under single bandwidth processing a single sequence. Local maximizers of the diagnostic statistic or the corresponding minimum p-values within bandwidth h are identified.

Usage

```
SARAp(Y, h, hh = 2 * h, sigma = NULL)
```

Arguments

`Y`	the numeric vector of the intensities of markers
`h`	the bandwidth for the screening procedure, defaults to 10
`hh`	the bandwidth for the local minimum procedure
`sigma`	the standard deviation for the intensities between two adjacent change points, defaults to NULL

Value

The return is a vector of local p-value minimum and the marker index of these minimum.

`index`	numeric vector a position index for all the local p-value minimum
`pV`	local minimum p-values

See Also

estimateSigma for estimation of the standard deviation between two adjacent change-points. localMax for calculation of the local maximizers of local diagnostic function

---

`smooth` *Smooth the original intensities to remove outliers*

Description

This function runs the smoothing procedure in the original intensities to remove outliers.

Usage

```
smooth(lrr, R = 10, t = 2)
```

Arguments

`lrr`	the matrix of the signal intensities. Each column presents a sequence or subject, each row represents a single marker
`t`	the tuning parameter for smoothing region, defaults to 2

Examples

```
# Input the example data of SNP genotyping data from Affymatrix Human SNP Array 6.0 platform.
data(example.data.lrr)
lrr <-example.data.lrr
lrr.smo <- smooth(lrr, R = 10, t = 2)
```

Index