

# Package ‘cgh’

February 14, 2012

**Version** 1.0-7.1

**Date** 2009-11-20

**Title** Microarray CGH analysis using the Smith-Waterman algorithm

**Author** Tom Price <t0mpr1c3@gmail.com>

**Maintainer** Tom Price <t0mpr1c3@gmail.com>

**Depends** R (>= 2.6.0)

**Description** Functions to analyze microarray comparative genome hybridization data using the Smith-Waterman algorithm

**LazyLoad** Yes

**License** GPL (>= 2)

**URL** <http://sgdp.iop.kcl.ac.uk/tprice/software.html>

**Repository** CRAN

**Date/Publication** 2010-05-07 16:10:28

## R topics documented:

sw . . . . .	2
sw.perm.test . . . . .	3
sw.plot . . . . .	4
sw.rob . . . . .	6
sw.threshold . . . . .	7
<b>Index</b>	<b>9</b>

---

`sw`*Perform the Smith-Waterman Algorithm*

---

**Description**

Perform the Smith-Waterman algorithm on a vector of real values.

**Usage**

```
sw(x, max.nIslands = NULL, trace = FALSE)
```

**Arguments**

<code>x</code>	a vector of real values
<code>max.nIslands</code>	the number of iterations of the algorithm performed. Each iteration finds the next highest-scoring 'island' of positive values. Set to NULL to find all islands
<code>trace</code>	print verbose output if TRUE

**Details**

The Smith-Waterman algorithm detects 'islands' of positive scores in a vector of real values. The input values should have a negative mean. The algorithm can be used to identify regions of copy number change in microarray fluorescence logratios, once the logratios have been adjusted for sign and a suitable threshold value subtracted to ensure a negative mean: see [sw.threshold](#)

**Value**

<code>x</code>	the input vector
<code>s</code>	a numeric vector containing the partial sums after one iteration of the Smith-Waterman algorithm
<code>score</code>	a numeric vector of island scores
<code>start</code>	a numeric vector of indices identifying the start of each island
<code>length</code>	a numeric vector of island lengths

**Author(s)**

T.S.Price

**References**

Smith TF, Waterman MS. Identification of common molecular subsequences. *J Mol Biol.* 1981;147(1):195-7.

Price TS, et al. SW-ARRAY: a dynamic programming solution for the identification of copy-number changes in genomic DNA using array comparative genome hybridization data. *Nucl Acids Res.* 2005;33(11):3455-3464.

**See Also**

[sw.threshold](#) [sw.perm.test](#) [sw.rob](#) [sw.plot](#)

**Examples**

```
## simulate vector of logratios
set.seed(3)
logratio <- c(rnorm(20) - 1, rnorm(20))

## invert sign of values and subtract threshold to ensure negative mean
x <- sw.threshold(logratio, function(x) median(x) + .2 * mad(x), sign = -1)

## perform Smith-Waterman algorithm
sw(x, trace = TRUE)
```

---

sw.perm.test

*Permutation Test for Smith-Waterman Algorithm*

---

**Description**

Perform a permutation test of island scores from the Smith-Waterman algorithm.

**Usage**

```
sw.perm.test(x, max.nIslands = 1, nIter = 1000, seed = NULL, trace = FALSE)
```

**Arguments**

x	a vector of real values
max.nIslands	number of iterations of the algorithm, each iteration finding the next highest-scoring 'island' of positive values, or NULL to find all islands
nIter	number of permutations of the input data used in the test
seed	seed for the random number generator, or NULL to use a faster random number generator that cannot be seeded
trace	print verbose output if TRUE

**Value**

A vector of probability values, calculated as the proportion of instances for which performing the Smith-Waterman algorithm on random permutations of the data identifies a higher-scoring island than the islands identified when the algorithm is performed on the original data

**Author(s)**

T.S.Price

## References

Price TS, et al. SW-ARRAY: a dynamic programming solution for the identification of copy-number changes in genomic DNA using array comparative genome hybridization data. *Nucl Acids Res.* 2005;33(11):3455-3464.

## See Also

[sw](#)

## Examples

```
## simulate vector of logratios
set.seed(3)
logratio <- c(rnorm(20) - 1, rnorm(20))

## invert sign of values and subtract threshold to ensure negative mean
x <- sw.threshold(logratio, function(x) median(x) + .2 * mad(x), -1)

## perform Smith-Waterman
sw(x)

## perform permutation test on the islands identified
sw.perm.test(x, max.nIslands = NULL, nIter= 1e4)
```

---

sw.plot

*Plot Results of Smith-Waterman Algorithm*

---

## Description

This function plots the sign-adjusted logratios by their chromosomal location. It can superimpose the location of the highest-scoring island found by the Smith-Waterman algorithm, the results of a robustness analysis, and the expected logratios based on known copy numbers in the test DNA.

## Usage

```
sw.plot(logratio, location = seq(length(logratio)),
        threshold.func = function(x) median(x) + .2 * mad(x),
        sign = -1, highest = TRUE, expected = NULL, rob = NULL, legend = TRUE,
        xlab = "Chromosomal location", ylab = "Intensity log ratio", ...)
```

## Arguments

logratio            a vector of logratios, not adjusted for sign or threshold  
location            a vector of chromosomal locations corresponding to the log ratios  
threshold.func    threshold function: see [sw.threshold](#)

sign	sign of logratio adjustment: see <a href="#">sw.threshold</a>
highest	plot location of highest-scoring island if TRUE
expected	a vector of expected copy numbers, or NULL
rob	a vector of robustness scores, or NULL
legend	plot legend if TRUE
xlab	X axis label
ylab	Y axis label
...	other arguments passed to the 'plot' function

### Author(s)

T.S.Price

### References

Price TS, et al. SW-ARRAY: a dynamic programming solution for the identification of copy-number changes in genomic DNA using array comparative genome hybridization data. *Nucl Acids Res.* 2005;33(11):3455-3464.

### See Also

[sw](#) [sw.threshold](#) [sw.perm.test](#) [sw.rob](#)

### Examples

```
## simulate vector of logratios
set.seed(3)
logratio <- c(rnorm(20) - 1, rnorm(20))

## invert sign of values and subtract threshold to ensure negative mean
x <- sw.threshold(logratio, function(x) median(x) + .2 * mad(x), -1)

## perform permutation test for islands identified
p <- sw.perm.test(x, max.nIslands = NULL, nIter = 1e4)

## calculate robustness scores
r <- sw.rob(x)

## plot results
sw.plot(logratio, seq(length(logratio)),
        function(x) median(x) + .2 * mad(x), sign = -1, rob = r,
        main = paste("Toy dataset, highest-scoring island p =", p[1]))
```

---

`sw.rob`*Robustness Calculation for Smith-Waterman Algorithm*

---

**Description**

Calculate robustness scores to evaluate how sensitive to the threshold value is the localisation of the highest-scoring island identified by the Smith-Waterman algorithm

**Usage**

```
sw.rob(x, lo.func = function(x) median(x),  
      hi.func = function(x) median(x) + .4 * mad(x), prec = 100)
```

**Arguments**

<code>x</code>	a vector of real values
<code>lo.func</code>	a function for the lowest threshold value
<code>hi.func</code>	a function for the highest threshold value
<code>prec</code>	the precision of the calculation.

**Details**

This function performs a sensitivity analysis to determine the robustness the localisation of the highest-scoring island obtained by the Smith-Waterman algorithm to different values of the threshold. The Smith-Waterman algorithm is run repeatedly, each time using a different threshold value. The range of threshold values used is that obtained by dividing ( `lo.func(x)`, `hi.func(x)` ) into 'prec' equal intervals. The robustness is calculated as the proportion of times that a particular chromosomal location falls within the highest-scoring island.

**Value**

A vector of robustness values equal in length to the input vector.

**Author(s)**

T.S.Price

**References**

Price TS, et al. SW-ARRAY: a dynamic programming solution for the identification of copy-number changes in genomic DNA using array comparative genome hybridization data. *Nucl Acids Res.* 2005;33(11):3455-3464.

**See Also**

[SW](#)

**Examples**

```
## simulate vector of logratios
set.seed(3)
logratio <- c(rnorm(20) - 1, rnorm(20))

## invert sign of values and subtract threshold to ensure negative mean
x <- sw.threshold(logratio, function(x) median(x) + .2 * mad(x), -1)

## calculate robustness values
sw.rob(x)
```

---

sw.threshold	<i>Threshold function</i>
--------------	---------------------------

---

**Description**

Function to adjust intensity logratios for sign and threshold before performing the Smith-Waterman Algorithm.

**Usage**

```
sw.threshold(logratio,
             threshold.func = function(x) median(x) + .2 * mad(x), sign = +1)
```

**Arguments**

logratio	a vector of real values, corresponding to fluorescence intensity logratios
threshold.func	function for calculating threshold
sign	sign of logratio adjustment

**Details**

The purpose of this function is to adjust the microarray fluorescence intensity logratios to ensure that they have the appropriate sign and a mean that is less than zero. `sign = +1` is used to detect polysomy (regions of copy number change increase) in test:control logratios. Conversely, `sign = -1` is used – inverting the sign of the logratios – to detect deletions (regions of copy number decrease). A threshold, calculated using the threshold function, is subtracted from the sign-adjusted logratios to ensure that they have a negative mean. The default threshold function is equal to the median, plus a small constant multiplied by a robust estimator of the standard deviation.

**Value**

A numeric vector equal to  
`sign * logratio - threshold.func( sign * logratio )`

**Author(s)**

T.S.Price

**References**

Price TS, et al. SW-ARRAY: a dynamic programming solution for the identification of copy-number changes in genomic DNA using array comparative genome hybridization data. Nucl Acids Res. 2005;33(11):3455-3464.

**See Also**

[SW](#)

**Examples**

```
## simulate vector of logratios
set.seed(3)
logratio <- c(rnorm(20) - 1, rnorm(20))

## invert sign of values and subtract threshold to ensure negative mean
x <- sw.threshold(logratio, function(x) median(x) + .2 * mad(x), sign = -1)

## perform Smith-Waterman algorithm
sw(x, trace = TRUE)
```

# Index

## \*Topic **misc**

- sw, [2](#)
- sw.perm.test, [3](#)
- sw.plot, [4](#)
- sw.rob, [6](#)
- sw.threshold, [7](#)

  

- sw, [2](#), [4–6](#), [8](#)
- sw.perm.test, [3](#), [3](#), [5](#)
- sw.plot, [3](#), [4](#)
- sw.rob, [3](#), [5](#), [6](#)
- sw.threshold, [2–5](#), [7](#)