# Package 'changepointsR'

January 26, 2023

**Title** Post-Selection Inference for Changepoints

**Version** 0.0.0.9000

**Description** Implement post-
selection inference for change in mean models, for several changepoint algorithms.

**License** GPL (>= 3)

**Encoding** UTF-8

**Roxygen** list(markdown = TRUE)

**RoxygenNote** 7.2.3

**Suggests** knitr,
rmarkdown,
changepoint,
ggplot2

**VignetteBuilder** knitr

**Imports** ChangepointInference

## R topics documented:

binary_segmentation     *Binary segmentation*

### Description

Binary segmentation algorithm for detecting changepoints, in the change in mean model. The algorithm terminates when either

1. the maximum number of changepoints has been found; or

2. there are no remaining points where the CUSUM statistic is above the threshold.

### Usage

```
binary_segmentation(y, threshold = NULL, maxiter = NULL)
```

### Arguments

| | |
|---|---|
| y | Numeric vector of data. |
| threshold | Threshold for determining changepoint candidates; defaults to `sqrt(2*log(n)*var(y))`. |
| maxiter | Maximum number of changepoints to find; defaults to `length(y) -1`. |

### Value

A list:

- `results` Dataframe containing results
- `changepoints` Vector of changepoints detected
- `threshold` Value of `threshold`
- `maxiter` Value of `maxiter`

### Examples

```
set.seed(10)
y <- rnorm(100) + c(rep(1,20), rep(-1,30), rep(1,50))
binary_segmentation(y)
```

calculate_interval     *Calculate interval*

### Description

Wrapper function for `calculate_interval_bs`, `calculate_interval_wbs`, and `calculate_interval_not`.

## Usage

```
calculate_interval(
  y,
  method,
  results,
  nu,
  nu2 = NULL,
  nuTy = NULL,
  n.cp = NULL
)
```

## Arguments

| | |
|---|---|
| y | Numeric vector of data |
| method | Character string; "bs" for binary segmentation; "wbs" for wild binary segmentation; "not" for narrowest over threshold. |
| results | Output of changepoint algorithm (`binary_segmentation`, `wild_binary_segmentation`, or `narrowest_over_threshold`). |
| nu | Numeric vector. |
| nu2 | Value of $\|\nu\|_2^2$. |
| nuTy | Value of $\nu^T y$. |
| n.cp | Maximum number of changepoints to detect. |

## Value

A 2-dimensional vector.

## Examples

```
set.seed(100)
y <- rnorm(100) + c(rep(1,50), rep(-1,50))
results <- binary_segmentation(y, threshold=4)
b <- results$results$b[ results$results$cp==1 ]
h <- 10
nu <- c(rep(0, b[1]-h), rep(1/h, h), rep(-1/h, h), rep(0, length(y)-b[1]-h))
calculate_interval(y, "bs", results, nu)
```

---

calculate_interval_bs    *Calculate interval for binary segmentation*

---

## Description

Find values of phi which satisfy the required inequalities so that applying binary segmentation to $y'(\phi)$ returns (b,d). Option to only consider part of (b,d), e.g. if we want b[1] to be in the set of detected changepoints but are not concerned about other changepoints.

## Usage

```
calculate_interval_bs(
  y,
  nu,
  b,
  d,
  nu2 = NULL,
  nuTy = NULL,
  threshold = NULL,
  n.cp = NULL
)
```

## Arguments

| | |
|---|---|
| y | Numeric vector of data. |
| nu | Numeric vector. |
| b | Vector of changepoints detected by binary segmentation algorithm. |
| d | Directions of changepoints detected by binary segmentation algorithm. Vector whose entries are all equal to 1 or -1. |
| nu2 | Value of $||\nu||_2^2$; optional. |
| nuTy | Value of $\nu^T y$; optional. |
| threshold | Threshold used in binary segmentation algorithm. |
| n.cp | Maximum number of changepoints to detect of binary segmentation algorithm. |

## Details

Used inside `calculate_S` if `method = "bs"`.

## Value

A 2-dimensional vector.

## Examples

```
set.seed(100)
y <- rnorm(100) + c(rep(1,50), rep(-1,50))
results <- binary_segmentation(y, threshold=4)
b <- results$results$b[ results$results$cp==1 ]
d <- results$results$d[ results$results$cp==1 ]
h <- 10
nu <- c(rep(0, b[1]-h), rep(1/h, h), rep(-1/h, h), rep(0, length(y)-b[1]-h))
calculate_interval_bs(y, nu, b, d, threshold=4)
```

```
calculate_interval_not
```
*Calculate interval for narrowest over threshold*

**Description**

Find values of phi which satisfy the required inequalities so that applying narrowest over threshold to $y'(\phi)$ returns (b,d).

**Usage**

```
calculate_interval_not(y, nu, results, nu2 = NULL, nuTy = NULL)
```

**Arguments**

| | |
|---|---|
| y | Numeric vector of data. |
| nu | Numeric vector. |
| results | Output of narrowest_over_threshold. |
| nu2 | Value of $||\nu||_2^2$. |
| nuTy | Value of $\nu^T y$. |

**Details**

Used inside calculate_S if method = "not".

**Value**

A 2-dimensional vector.

**Examples**

```
set.seed(100)
y <- rnorm(100) + c(rep(1,50), rep(-1,50))
results <- narrowest_over_threshold(y, lambda=4, N=50)
b <- results$results$b
h <- 10
nu <- c(rep(0, b[1]-h), rep(1/h, h), rep(-1/h, h), rep(0, length(y)-b[1]-h))
calculate_interval_not(y, nu, results=results)
```

calculate_interval_wbs

*Calculate interval for wild binary segmentation*

### Description

Find values of phi which satisfy the required inequalities so that applying wild binary segmentation to $y'(\phi)$ returns (b,d). Option to only consider part of (b,d), e.g. if we want b[1] to be in the set of detected changepoints but are not concerned about other changepoints.

### Usage

```
calculate_interval_wbs(
  y,
  nu,
  results = NULL,
  b = NULL,
  d = NULL,
  s = NULL,
  e = NULL,
  rand_ints = NULL,
  nu2 = NULL,
  nuTy = NULL,
  threshold = NULL,
  n.cp = NULL
)
```

### Arguments

| | |
|---|---|
| y | Numeric vector of data. |
| nu | Numeric vector. |
| results | Output of `wild_binary_segmentation`. |
| b | Numeric vector of changepoints. Ignored if `results` specified. |
| d | Numeric vector containing directions of changepoints; all entries should be either +1 or -1. Ignored if `results` specified. |
| s | Numeric vector containing starting indices of changepoint-containing intervals. Ignored if `results` specified. |
| e | Numeric vector containing ending indices of changepoint-containing intervals. Ignored if `results` specified. |
| rand_ints | Matrix containing starting and ending points of random intervals used for wild binary segmentation algorithm. Ignored if `results` specified. |
| nu2 | Value of $\|\nu\|_2^2$. |
| nuTy | Value of $\nu^T y$. |
| threshold | Changepoint threshold used in wild binary segmentation algorithm. Ignored if `results` specified. |
| n.cp | Maximum number of changepoints to detect. |

**Details**

Used inside `calculate_S` if `method = "wbs"`.

**Value**

A 2-dimensional vector.

**Examples**

```
set.seed(100)
y <- rnorm(100) + c(rep(1,50), rep(-1,50))
results <- wild_binary_segmentation(y, threshold=4, num_rand_samples=50)
b <- results$results$b
h <- 10
nu <- c(rep(0, b[1]-h), rep(1/h, h), rep(-1/h, h), rep(0, length(y)-b[1]-h))
calculate_interval_wbs(y, nu, results=results)
```

---

calculate_pvals          *Calculate p-values for change in mean model*

---

**Description**

Calculate p-values for detected changepoints, for the change in mean model. Changepoints can be detected using either binary segmentation, wild binary segmentation, or narrowest over threshold.

**Usage**

```
calculate_pvals(
  y,
  method = "bs",
  results = NULL,
  N = 10,
  threshold = NULL,
  maxiter = NULL,
  h = NULL,
  cp_bound = TRUE,
  sigma2 = 1,
  eps0 = 0.01,
  include_original = TRUE,
  num_pvals = NULL,
  random_samples = NULL,
  num_rand_samples = NULL,
  seeded = FALSE,
  decay = NULL,
  return_probs = FALSE
)
```

**Arguments**

| | |
|---|---|
| `y` | Numeric vector of data. |
| `method` | Method used to generate changepoints. Options: `"bs"` for binary segmentation, `"wbs"` for wild or seeded binary segmentation, `"not"` for narrowest over threshold. |
| `results` | Output of changepoint algorithm (either `binary_segmentation`, `wild_binary_segmentation`, or `narrowest_over_threshold`). |
| `N` | Positive integer. Number of random $\psi$'s to generate; defaults to 10. |
| `threshold` | Minimum threshold for changepoint detection. Ignored if `results` is specified. |
| `maxiter` | Positive integer. Maximum number of changepoints to find in changepoint algorithm. Ignored if `results` is specified |
| `h` | Positive integer (>=2) or `NULL`. Window size. If `NULL` then the changepoints either side of the changepoint of interest are used to define the window. |
| `cp_bound` | Logical. ... |
| `sigma2` | Variance of `y`. |
| `eps0` | Hyperparameter for calculating S. |
| `include_original` | |
| | Logical. Whether to include the $\psi$ value corresponding to the observed data in place of one of the random samples. Defaults to `TRUE`. |
| `num_pvals` | Integer or NA. Maximum number of p-values to calculate; if set to NA, it will default to `length(y) -1`. If the number of changepoints detected by the binary segmentation algorithm is less than or equal to `num_pvals`, then p-values will be calculated for all changepoints. If `num_pvals` is less than the number of changepoints detected, p-values will only be calculated for the first `num_pvals` changepoints, in order of detection. |
| `random_samples` | Matrix containing random intervals. For wild binary segmentation and narrowest over threshold only. Ignored if `results` specified or `method = "bs"`. |
| `num_rand_samples` | |
| | Number of random intervals to use for wild binary segmentation or narrowest over threshold. Ignored if `random_samples` or `results` is specified, `method = "bs"`, or `seeded = TRUE`. |
| `seeded` | Logical. If `TRUE`, use seeded binary segmentation rather than wild binary segmentation. Only used if `method = "wbs"`. |
| `decay` | Decay parameter for seeded binary segmentation. Only used if `method = "wbs"` and `seeded = TRUE`. |
| `return_probs` | Logical. If `TRUE`, the values of $Pr(\phi \in S \& |\phi| > |\phi_{obs}|)$ and $Pr(\phi \in S)$ for each $\psi$ will be included in the output. |

**Details**

Given a changepoint of interest $\tau_j$, there are two options for the null hypothesis:

- There are no changepoints within a window size $h$ of $\tau_j$. In this case `h` should be supplied. If `cp_bound` is `TRUE`, then if there is a another estimated changepoint within $h$ of $\tau_j$, this will be used.
- There are no other changepoints within ...
- There are no other changepoints between $\tau_{j-1}$ and $\tau_{j+1}$. In this case `h` should be set to `NULL`.

## Value

A list.

- p_value A vector of p-values
- P_both If return_probs=TRUE, a matrix containing values of $Pr(\phi \in S \& |\phi| > |\phi_{obs}|)$ for each $\psi$; otherwise NA.
- P_phi_in_S If return_probs=TRUE, a matrix containing values of $Pr(\phi \in S)$ for each $\psi$; otherwise NA.

## Examples

```
set.seed(100)
y <- rnorm(200) + c(rep(1,50), rep(-1,50), rep(1,50), rep(-1,50))
results <- binary_segmentation(y, threshold=4)
calculate_pvals(y, method="bs", results=results, h=10)

y <- rnorm(200) + c(rep(1,50), rep(-1,50), rep(1,50), rep(-1,50))
results <- binary_segmentation(y, threshold=4)
calculate_pvals(y, method="bs", results=results, h=10)
```

---

calculate_S                 *Calculate S*

---

## Description

A function for calculating the set of interest S, when the changepoint algorithm used is one of binary segmentation, wild binary segmentation, or narrowest over threshold.

## Usage

```
calculate_S(
  y,
  nu,
  results = NULL,
  b = NULL,
  d = NULL,
  threshold = NULL,
  maxiter = NULL,
  nu2 = NULL,
  nuTy = NULL,
  eps0 = 0.01,
  first_cp_only = FALSE,
  method = "bs",
  rand_ints = NULL,
  seeded = FALSE,
  decay = NULL
)
```

## Arguments

| | |
|---|---|
| y | Numeric vector of data. |
| nu | Numeric vector. |
| results | Output of changepoint algorithm (either `binary_segmentation`, `wild_binary_segmentation`, or `narrowest_over_threshold`. |
| b | Vector of changepoints; ignored if `results` specified. |
| d | Vector containing directions of changepoints (+1 or -1); ignored if `results` specified. |
| threshold | Threshold for detecting changepoints; ignored if `results` specified. |
| maxiter | Maximum number of changepoints to detect; ignored if `results` specified. |
| nu2 | Value of $\|\|\nu\|\|_2^2$; optional. |
| nuTy | Value of $nu^T y$; optional. |
| eps0 | Hyperparameter. |
| first_cp_only | Logical. If `TRUE`, condition on the fact that the changepoint of interest is in the model; if `FALSE`, condition on all changepoints. Defaults to `FALSE`. |
| method | Character. One of `"bs"` (binary segmentation), `"wbs"` (wild binary segmentation), or `"not"` (narrowest over threshold). Defaults to `"bs"`. |
| rand_ints | Matrix containing random intervals for changepoint algorithm. Ignored if `results` specified or `method = "bs"`. |
| seeded | Logical. For `method = "wbs"` only, whether to use seeded binary segmentation. |
| decay | Decay parameter for seeded binary segmentation. Only used if `method = "wbs"` and `seeded = TRUE`. |

## Value

A dataframe containing intervals with the changepoints obtained when $\phi$ is in each interval.

## Examples

```
set.seed(100)
y <- rnorm(100) + c(rep(1,50), rep(-1,50))
results <- binary_segmentation(y)
print(results$changepoints)
h <- 10
nu <- c(rep(0, results$changepoints[1] - h), rep(1/h, h), rep(-1/h, h), rep(0, length(y) - results$changepoints
calculate_S(y, nu, results, method="bs", first_cp_only=TRUE)
```

---

| cusum | *Calculate CUSUM statistics for a vector* |
|---|---|

---

## Description

Calculate CUSUM statistics for a vector y.

## Usage

```
cusum(y, s = 1, e = length(y), return_full = FALSE)
```

## Arguments

| | |
|---|---|
| y | A numeric vector. |
| s | Starting index. |
| e | Ending index. |
| return_full | Logical. If TRUE, returns vector of same length as y, with indices outside of (s,e -1) set to NA. If FALSE, a vector of length e -s + 1 is returned. |

## Value

A numeric vector.

## Examples

```
y <- c(1, 3, 2, 5, 3)
cusum(y)
```

---

| cusum_phi_vec | *Calculate vector of CUSUM statistics for data y, in terms of phi* |
|---|---|

---

## Description

Used inside `calculate_pvals`.

## Usage

```
cusum_phi_vec(y, nu, nu2 = NULL, nuTy = NULL, s = 1, e = length(y))
```

## Arguments

| | |
|---|---|
| y | Numeric vector of data. |
| nu | Numeric vector. |
| nu2 | Numeric; value of the squared 2-norm of nu. |
| nuTy | Value of nu^T y. |
| s | Starting point for calculating CUSUM statistics, defaults to 1. |
| e | Ending point for calculating CUSUM statistics, degaults to `length(y)`. |

## Value

Returns `length(y)` x 2 matrix such that `cusum_phi_vec(y,nu)^T %*% c(1,phi) = cusum(y_phi(y,nu,phi))`.

## Examples

```
set.seed(100)
y <- rnorm(100) + c(rep(1,40), rep(-1,60))
results <- binary_segmentation(y)
b <- results$changepoints[1]
nu <- c(rep(1/b, b), rep(-1/(100-b), 100-b))
cusum_phi_vec(y, nu)
```

---

find_cps                              *Find changepoints.*

---

### Description

Wrapper function for `binary_segmentation`, `wild_binary_segmentation`, and `narrowest_over_threshold`.

### Usage

```
find_cps(
  y,
  method,
  threshold = NULL,
  maxiter = NULL,
  num_rand_ints = NULL,
  rand_ints = NULL,
  seeded = FALSE,
  decay = NULL
)
```

### Arguments

| | |
|---|---|
| y | Numeric vector of data. |
| method | Character string: `"bs"` for binary segmentation; `"wbs"` for wild binary segmentation; `"not"` for narrowest over threshold. |
| threshold | Numeric; changepoint detection threshold for CUSUM statistic. |
| maxiter | Integer; maximum number of changepoints to detect. |
| num_rand_ints | Integer; number of random intervals. Ignored if `rand_ints` is specified, or if `method = "bs"`. |
| rand_ints | Matrix containing random intervals for wild binary segmentation or narrowest over threshold. Ignored if `method = "bs"`. |
| seeded | Logical; if `TRUE` and `method = "wbs"`, then seeded binary segmentation is implemented. |
| decay | Decay parameter for seeded binary segmentation; only used if `method = "wbs"` and `seeded = TRUE`. |

### Value

A list:

- `results` Dataframe containing results
- `changepoints` Vector of changepoints detected
- `rand_ints` (Not if `method = "bs"`) Matrix containing random intervals used
- `threshold` Value of `threshold`
- `maxiter` Value of `maxiter`

## Examples

```
set.seed(100)
y <- rnorm(100) + c(rep(1,45), rep(-1,10), rep(1,45))
results_bs <- find_cps(y, "bs", threshold=4)
print(results_bs$results)

results_wbs <- find_cps(y, "wbs", threshold=4, num_rand_ints=100)
print(results_wbs$results)

results_not <- find_cps(y, "not", threshold=4, num_rand_ints=100)
print(results_not$results)
```

---

generate_random_intervals

*Generate random intervals*

---

## Description

Generate random intervals within the range (1,n). For use in wild binary segmentation and narrowest over threshold algorithms.

## Usage

```
generate_random_intervals(n, N, min_width = 2)
```

## Arguments

| | |
|---|---|
| n | Integer; upper limit |
| N | Integer; number of intervals to generate |
| min_width | Integer; minimum width of interval allowed |

## Details

If the number of intervals supplied is greater than the total number of possible unique intervals contained in $[1, n]$ with width at least min_width, then the output will be a list of all possible intervals contained in $[1, n]$.

## Value

An N x 2 matrix of integers

## Examples

```
generate_random_intervals(100, 10, 2)
```

---

l0_segmentation_psi          *Post-selection inference for L0 segmentation*

---

### Description

Post-selection inference for L0 segmentation. This uses the functions `changepoint_estimates` and `changepoint_inference` from the package `ChangepointInference`.

### Usage

```
l0_segmentation_psi(
  y,
  lambda,
  N,
  h,
  sigma2 = 1,
  sig = 4,
  include_original = TRUE,
  num_pvals = NULL
)
```

### Arguments

| | |
|---|---|
| y | Numeric vector of data |
| lambda | Threshold parameter |
| N | Number of samples of psi to take |
| h | Window size |
| sigma2 | Variance of y |
| sig | Tuning parameter |
| include_original | |
| | Logical; whether to include observed value as psi in place as one of the random samples; defaults to `TRUE` |
| num_pvals | Maximum number of p-values to calculate |

### Value

A list:

- b Vector of changepoints
- `p_value` Vector of p-values
- `p_value_orig` Vector of p-values obtained using fixed $\psi = \psi_{obs}$
- `nuTy` Value of $\nu^T y$
- `P_both` Matrix containing values of $Pr(|\phi| > |\phi_{obs}| \& \phi \in S)$
- `P_phi_in_S` Matrix containing values of $Pr(\phi \in S)$
- `P_both_orig` Vector containing values of $Pr(|\phi| > |\phi_{obs}| \& \phi \in S)$ for fixed $\psi = \psi_{obs}$
- `P_phi_in_S_orig` Vector containing values of $Pr(\phi \in S)$ for fixed $\psi = \psi_{obs}$

## Examples

```
set.seed(100)
y <- rnorm(100) + c(rep(1,40), rep(-1,20), rep(1,40))
l0_segmentation_psi(y, 4, 10, 10)
```

---

narrowest_over_threshold

*Narrowest over threshold for change in mean.*

---

## Description

Apply the narrowest over threshold changepoint algorithm to a vector of data. The model is the piecewise constant mean model.

## Usage

```
narrowest_over_threshold(y, lambda, rand_ints = NULL, N = 1000, max_cps = NULL)
```

## Arguments

| | |
|---|---|
| y | A numeric vector of data. |
| lambda | Threshold for CUSUM statistic; a scalar. |
| rand_ints | N x 2 matrix containing random intervals; optional. |
| N | Number of random intervals; defaults to 1000. Ignored if `rand_ints` is specified. |
| max_cps | Maximum number of changepoints to return; defaults to `length(y) -1` if NULL. |

## Value

A list.

- results Dataframe containing results
- changepoints Vector of changepoints detected
- rand_ints N x 2 matrix containing random intervals used in the algorithm.
- threshold Value of `lambda`
- maxiter Value of `maxiter`

## Examples

```
y <- rnorm(200) + c(rep(1,90), rep(-1,20), rep(1,90))
lambda <- 4
results <- narrowest_over_threshold(y, lambda, N=500)
print(results$results)
print(results$changepoints)
```

---

wild_binary_segmentation

*Wild Binary Segmentation*

---

**Description**

Implements wild binary segmentation algorithm for changepoint detection.

**Usage**

```
wild_binary_segmentation(
  y,
  num_rand_samples = 1000,
  random_samples = NULL,
  threshold = NULL,
  maxiter = NULL,
  seeded = FALSE,
  decay = sqrt(2)
)
```

**Arguments**

| | |
|---|---|
| y | Numeric vector of data. |
| num_rand_samples | |
| | Number of random intervals to use. Ignored if `random_samples` is supplied. |
| random_samples | N x 2 matrix of random intervals. |
| threshold | Minimum threshold for determining changepoint candidates. |
| maxiter | Maximum number of changepoints to find. |
| seeded | Logical; if TRUE, seeded binary segmentation is used. |
| decay | Decay parameter for seeded binary segmentation. Only used if seeded = TRUE. |

**Value**

A list.

- `results` Dataframe containing results
- `changepoints` Vector of changepoints detected
- `rand_ints` N x 2 matrix containing random intervals used in the algorithm.
- `threshold` Value of `threshold`
- `maxiter` Value of `maxiter`

**Examples**

```
set.seed(100)
y <- rnorm(100) + c(rep(1,40), rep(0,10), rep(1,50))
results <- wild_binary_segmentation(y, num_rand_samples=500)
print(results$results)
```

| y_phi | *Calculate y'(phi)* |
|---|---|

### Description

Calculate $y'(\phi) = y - (||\nu||_2^2)^{-1}\nu\nu^T y + (||\nu||_2^2)^{-1}\nu\phi$.

### Usage

```
y_phi(y, nu, phi, nu2 = NULL, nuTy = NULL)
```

### Arguments

| | |
|---|---|
| y | Numeric data vector. |
| nu | Numeric vector of same length as y. |
| phi | Numeric. |
| nu2 | Value of $||\nu||_2^2$; optional. |
| nuTy | Value of $\nu^T y$; optional. |

### Value

Numeric vector.

### Examples

```
y <- rnorm(20) + c(rep(1, 10), rep(-1, 10))
nu <- c(rep(0, 5), rep(1/5, 5), rep(-1/5, 5), rep(0, 5))
phi <- 2
y_phi(y, nu, phi)
```

# Index