Package 'stcpR6'

September 8, 2024

Title Sequential Test and Change-Point Detection Algorithms Based on E-Values / E-Detectors

Version 0.9.7

Description Algorithms of nonparametric sequential test and online change-point detection for streams of univariate (sub-)Gaussian, binary, and bounded random variables, introduced in following publications - Shin et al. (2024) <doi:10.48550/arXiv.2203.03532>, Shin et al. (2021) <doi:10.48550/arXiv.2010.08082>.

License GPL (>= 3)

Encoding UTF-8

Imports methods, Rcpp (>= 1.0.12), R6

LinkingTo Rcpp

Suggests rmarkdown, knitr, testthat (>= 3.0.0)

Config/testthat/edition 3

URL https://github.com/shinjaehyeok/stcpR6

BugReports https://github.com/shinjaehyeok/stcpR6/issues

RoxygenNote 7.3.2

NeedsCompilation yes

Author Jaehyeok Shin [aut, cre] (<https://orcid.org/0000-0003-0464-915X>)

Maintainer Jaehyeok Shin <shinjaehyeok@gmail.com>

Repository CRAN

Date/Publication 2024-09-08 00:30:02 UTC

Contents

`stcpR6-package` *stcpR6: Sequential Test and Change-Point Detection Algorithms Based on E-Values / E-Detectors*

Description

Algorithms of nonparametric sequential test and online change-point detection for streams of univariate (sub-)Gaussian, binary, and bounded random variables, introduced in following publications - Shin et al. (2024) doi:10.48550/arXiv.2203.03532, Shin et al. (2021) doi:10.48550/arXiv.2010.08082.

Author(s)

Maintainer: Jaehyeok Shin <shinjaehyeok@gmail.com> (ORCID)

See Also

Useful links:

- https://github.com/shinjaehyeok/stcpR6
- Report bugs at https://github.com/shinjaehyeok/stcpR6/issues

---

`checkDeltaRange` *Check whether the input delta parameters are acceptable*

Description

For each method and family, check whether delta parameters are within expected range with respect to the pre-change parameter.

Usage

```
checkDeltaRange(method, family, alternative, m_pre, delta_lower, delta_upper)
```

Arguments

`method`	Method of the sequential procedure. • ST: Sequential test based on a mixture of E-values. • SR: Sequential change detection based on e-SR procedure. • CU: Sequential change detection based on e-CUSUM procedure. • GLRCU: Sequential change detection based on GLR-CUSUM procedure.
`family`	Distribution of underlying univariate observations. • Normal: (sub-)Gaussian with sigma = 1. • Ber: Bernoulli distribution on {0,1}. • Bounded: General bounded distribution on [0,1]
`alternative`	Alternative / post-change mean space • two.sided: Two-sided test / change detection • greater: Alternative /post-change mean is greater than null / pre-change one • less: Alternative /post-change mean is less than null / pre-change one
`m_pre`	The boundary of mean parameter in null / pre-change space
`delta_lower`	Minimum gap between null / pre-change space and alternative / post-change one. It must be strictly positive for ST, SR and CU. Currently, GLRCU does not support the minimum gap, and this param will be ignored.
`delta_upper`	Maximum gap between null / pre-change space and alternative / post-change one. It must be strictly positive for ST, SR and CU. Currently, GLRCU does not support the maximum gap, and this param will be ignored.

Value

A list of

1. Boolean indicating whether it is acceptable or not.
2. Character describing why it is not acceptable.
3. Updated delta_upper for the case where the original input was NULL

`compute_baseline` *Compute baseline processes.*

Description

Compute parameters to build baseline processes.

Usage

```
compute_baseline(
  alpha,
  delta_lower,
  delta_upper,
  psi_fn_list = generate_sub_G_fn(),
  v_min = 1,
  k_max = 200,
  tol = 1e-10
)
```

Arguments

alpha	ARL parameter in (0,1)
delta_lower	Lower bound of target Delta. It must be positive and smaller than or equal to delta_upper.
delta_upper	Upper bound of target Delta. It must be positive and larger than or equal to delta_lower.
psi_fn_list	A list of R functions that computes psi and psi_star functions. Can be generated by generate_sub_G_fn() or counterparts for sub_B and sub_E.
v_min	A lower bound of v function in the baseline process. Default is 1.
k_max	Positive integer to determine the maximum number of baselines. Default is 200.
tol	Tolerance of root-finding, positive numeric. Default is 1e-10.

Value

A list of 1. Parameters of baseline processes, 2. Mixing weights, 3. Auxiliary values for computation.

---

compute_baseline_for_sample_size

Compute baseline parameters given target variance process bounds.

Description

Given target variance process bounds for confidence sequences, compute baseline parameters.

Usage

```
compute_baseline_for_sample_size(
  alpha,
  v_upper,
  v_lower,
  psi_fn_list = generate_sub_G_fn(),
  v_min = 1,
```

```
  k_max = 200,
  tol = 1e-10
)
```

Arguments

alpha	ARL parameter in (0,1)
v_upper	Upper bound of the target variance process bound
v_lower	Lower bound of the target variance process bound.
psi_fn_list	A list of R functions that computes psi and psi_star functions. Can be generated by generate_sub_G_fn() or counterparts for sub_B and sub_E.
v_min	A lower bound of v function in the baseline process. Default is 1.
k_max	Positive integer to determine the maximum number of baselines. Default is 200.
tol	Tolerance of root-finding, positive numeric. Default is 1e-10.

Value

A list of 1. Parameters of baseline processes, 2. Mixing weights, 3. Auxiliary values for computation.

---

convertDeltaToExpParams

converted input deltas to parameters for exponential baselines

---

Description

For each exponential baseline family, convert delta range into corresponding lambdas and weights.

Usage

```
convertDeltaToExpParams(
  family,
  alternative,
  threshold,
  m_pre,
  delta_lower,
  delta_upper,
  k_max
)
```

Arguments

family	Distribution of underlying univariate observations. • Normal: (sub-)Gaussian with sigma = 1. • Ber: Bernoulli distribution on {0,1}. • Bounded: General bounded distribution on [0,1]
alternative	Alternative / post-change mean space • two.sided: Two-sided test / change detection • greater: Alternative /post-change mean is greater than null / pre-change one • less: Alternative /post-change mean is less than null / pre-change one
threshold	Stopping threshold. We recommend to use log(1/alpha) for "ST" and "SR" methods where alpha is a testing level or 1/ARL. for "CU" and "GRLCU", we recommend to tune the threshold by using domain-specific sampler to hit the target ARL.
m_pre	The boundary of mean parameter in null / pre-change space
delta_lower	Minimum gap between null / pre-change space and alternative / post-change one. It must be strictly positive for ST, SR and CU. Currently, GLRCU does not support the minimum gap, and this param will be ignored.
delta_upper	Maximum gap between null / pre-change space and alternative / post-change one. It must be strictly positive for ST, SR and CU. Currently, GLRCU does not support the maximum gap, and this param will be ignored.
k_max	Positive integer to determine the maximum number of baselines. For GLRCU method, it is used as the lookup window size for GLRCU statistics.

Value

A list of weights and lambdas

---

generate_sub_B_fn *Pre-defined psi_star functions for sub-Bernoulli family.*

Description

Pre-defined psi_star functions for sub-Bernoulli family.

Usage

```
generate_sub_B_fn(p = 0.5)
```

Arguments

p	The boundary of mean space of the pre-change distributions (default = 0.5).

Value

A list of pre-defined psi_star functions for sub-Bernoulli family.

generate_sub_E_fn *Pre-defined psi_star functions for sub-exponential family.*

Description

Pre-defined psi_star functions for sub-exponential family.

Usage

```
generate_sub_E_fn()
```

Value

A list of pre-defined psi_star functions for sub-exponential family.

generate_sub_G_fn *Pre-defined psi_star functions for sub-Gaussian family.*

Description

Pre-defined psi_star functions for sub-Gaussian family.

Usage

```
generate_sub_G_fn(sig = 1)
```

Arguments

sig	The sigma parameter of the sub-Gaussian family (default = 1).

Value

A list of pre-defined psi_star functions for sub-Gaussian family.

logSumExpTrick *log-sum-exp trick*

Description

Apply log-sum-exp trick to a numeric vector.

Usage

```
logSumExpTrick(xs)
```

Arguments

xs	A numeric vector.

Value

log of sum of exp of xs, which is equal to `log(sum(exp(xs)))`.

NormalCS *NormalCS Class*

Description

NormalCS class is used to compute always-valid confidence sequence for the standard normal process based on the STCP method.

Methods

Public methods:

- `NormalCS$new()`
- `NormalCS$print()`
- `NormalCS$getAlpha()`
- `NormalCS$getWeights()`
- `NormalCS$getLambdas()`
- `NormalCS$computeWidth()`
- `NormalCS$computeInterval()`

Method `new()`**:** Create a new NormalCS object.

Usage:

```
NormalCS$new(
  alternative = c("two.sided", "greater", "less"),
  alpha = 0.05,
  n_upper = 1000,
  n_lower = 1,
  weights = NULL,
  lambdas = NULL,
  k_max = 1000
)
```

Arguments:

`alternative` Alternative / post-change mean space

- two.sided: Two-sided test / change detection
- greater: Alternative /post-change mean is greater than null / pre-change one
- less: Alternative /post-change mean is less than null / pre-change one

`alpha` Upper bound on the type 1 error of the confidence sequence.

`n_upper` Upper bound of the target sample interval

`n_lower` Lower bound of the target sample interval

`weights` If not null, the input weights will be used to initialize the object instead of `n_upper` and `n_lower`.

`lambdas` If not null, the input lambdas will be used to initialize the object. instead of `n_upper` and `n_lower`.

`k_max` Positive integer to determine the maximum number of baselines.

Returns: A new `NormalCS` object.

Method `print()`**:** Print summary of Stcp object.

Usage:

```
NormalCS$print()
```

Method `getAlpha()`**:** Return the upper bound on the type 1 error

Usage:

```
NormalCS$getAlpha()
```

Method `getWeights()`**:** Return weights of mixture of e-values / e-detectors.

Usage:

```
NormalCS$getWeights()
```

Method `getLambdas()`**:** Return lambda parameters of mixture of e-values / e-detectors.

Usage:

```
NormalCS$getLambdas()
```

Method `computeWidth()`**:** Compute the width of confidence interval at time n.

Usage:

```
NormalCS$computeWidth(n)
```

Arguments:

n Positive time.

Method `computeInterval()`**:** Compute a vector of two end points of confidence interval at time n

Usage:

```
NormalCS$computeInterval(n, x_bar = 0)
```

Arguments:

n Positive time.

x_bar The center of the confidence interval.

Examples

```
# Initialize two-sided standard normal confidence sequence
# optimized for the interval [10, 100]
normal_cs <- NormalCS$new(
  alternative = "two.sided",
  alpha = 0.05,
  n_upper = 100,
  n_lower = 10
  )

# Compute confidence interval at n = 20 when observed sample mean = 0.5
normal_cs$computeInterval(20, x_bar = 0.5)

# (Advanced) NormalCS supports general variance process.
# Both n_upper and n_lower can be general positive numbers.
normal_cs2 <- NormalCS$new(
  alternative = "two.sided",
  alpha = 0.05,
  n_upper = 100.5,
  n_lower = 10.5
  )
# Confidence interval at n = 20.5
normal_cs$computeInterval(20.5, x_bar = 0.5)
```

`Stcp` *Stcp Class*

Description

Stcp class supports a unified framework for sequential tests and change detection algorithms for streams of univariate (sub-)Gaussian, binary, and bounded random variables.

Methods

Public methods:

- Stcp$new()
- Stcp$print()
- Stcp$getWeights()
- Stcp$getLambdas()
- Stcp$getLogValue()
- Stcp$getThreshold()
- Stcp$isStopped()
- Stcp$getTime()
- Stcp$getStoppedTime()
- Stcp$reset()
- Stcp$updateLogValues()
- Stcp$updateLogValuesUntilStop()
- Stcp$updateAndReturnHistories()
- Stcp$updateLogValuesByAvgs()
- Stcp$updateLogValuesUntilStopByAvgs()
- Stcp$updateAndReturnHistoriesByAvgs()

Method `new()`**:** Create a new Stcp object.

Usage:

```
Stcp$new(
  method = c("ST", "SR", "CU", "GLRCU"),
  family = c("Normal", "Ber", "Bounded"),
  alternative = c("two.sided", "greater", "less"),
  threshold = log(1/0.05),
  m_pre = 0,
  delta_lower = 0.1,
  delta_upper = NULL,
  weights = NULL,
  lambdas = NULL,
  k_max = 1000
)
```

Arguments:

`method` Method of the sequential procedure.

- ST: Sequential test based on a mixture of E-values.
- SR: Sequential change detection based on e-SR procedure.
- CU: Sequential change detection based on e-CUSUM procedure.
- GLRCU: Sequential change detection based on GLR-CUSUM procedure.

`family` Distribution of underlying univariate observations.

- Normal: (sub-)Gaussian with sigma = 1.
- Ber: Bernoulli distribution on {0,1}.
- Bounded: General bounded distribution on [0,1]

`alternative` Alternative / post-change mean space

- two.sided: Two-sided test / change detection
- greater: Alternative /post-change mean is greater than null / pre-change one
- less: Alternative /post-change mean is less than null / pre-change one

`threshold` Stopping threshold. We recommend to use log(1/alpha) for "ST" and "SR" methods where alpha is a testing level or 1/ARL. for "CU" and "GRLCU", we recommend to tune the threshold by using domain-specific sampler to hit the target ARL.

`m_pre` The boundary of mean parameter in null / pre-change space

`delta_lower` Minimum gap between null / pre-change space and alternative / post-change one. It must be strictly positive for ST, SR and CU. Currently, GLRCU does not support the minimum gap, and this param will be ignored.

`delta_upper` Maximum gap between null / pre-change space and alternative / post-change one. It must be strictly positive for ST, SR and CU. Currently, GLRCU does not support the maximum gap, and this param will be ignored.

`weights` If not null, the input weights will be used to initialize Stcp object.

`lambdas` If not null, the input lambdas will be used to initialize Stcp object.

`k_max` Positive integer to determine the maximum number of baselines. For GLRCU method, it is used as the lookup window size for GLRCU statistics.

Returns: A new `Stcp` object.

Method `print()`**:** Print summary of Stcp object.

Usage:

```
Stcp$print()
```

Method `getWeights()`**:** Return weights of mixture of e-values / e-detectors.

Usage:

```
Stcp$getWeights()
```

Method `getLambdas()`**:** Return lambda parameters of mixture of e-values / e-detectors.

Usage:

```
Stcp$getLambdas()
```

Method `getLogValue()`**:** Return the log value of mixture of e-values / e-detectors.

Usage:

```
Stcp$getLogValue()
```

Method `getThreshold()`**:** Return the threshold of the sequential test / change detection

Usage:

```
Stcp$getThreshold()
```

Method `isStopped()`**:** Return TRUE if the sequential test / change detection was stopped by crossing the threshold.

Usage:

```
Stcp$isStopped()
```

Method `getTime()`**:** Return the number of observations having been passed.

Usage:

```
Stcp$getTime()
```

Method `getStoppedTime()`**:** Return the stopped time. If it has been never stopped, return zero.

Usage:

```
Stcp$getStoppedTime()
```

Method `reset()`**:** Reset the stcp object to the initial setup.

Usage:

```
Stcp$reset()
```

Method `updateLogValues()`**:** Update the log value and related fields by passing a vector of observations.

Usage:

```
Stcp$updateLogValues(xs)
```

Arguments:

`xs` A numeric vector of observations.

Method `updateLogValuesUntilStop()`**:** Update the log value and related fields until the log value is crossing the boundary.

Usage:

```
Stcp$updateLogValuesUntilStop(xs)
```

Arguments:

`xs` A numeric vector of observations.

Method `updateAndReturnHistories()`**:** Update the log value and related fields then return updated log values by passing a vector of observations.

Usage:

```
Stcp$updateAndReturnHistories(xs)
```

Arguments:

`xs` A numeric vector of observations.

Method `updateLogValuesByAvgs()`**:** Update the log value and related fields by passing a vector of averages and number of corresponding samples.

Usage:

```
Stcp$updateLogValuesByAvgs(x_bars, ns)
```

Arguments:

`x_bars` A numeric vector of averages.

`ns` A numeric vector of sample sizes.

Method `updateLogValuesUntilStopByAvgs()`**:** Update the log value and related fields by passing a vector of averages and number of corresponding samples until the log value is crossing the boundary.

Usage:

```
Stcp$updateLogValuesUntilStopByAvgs(x_bars, ns)
```

Arguments:

`x_bars` A numeric vector of averages.

`ns` A numeric vector of sample sizes.

Method `updateAndReturnHistoriesByAvgs()`**:** Update the log value and related fields then return updated log values a vector of averages and number of corresponding samples.

Usage:

```
Stcp$updateAndReturnHistoriesByAvgs(x_bars, ns)
```

Arguments:

`x_bars` A numeric vector of averages.

`ns` A numeric vector of sample sizes.

Examples

```
# Sequential Normal mean test H0: mu <= 0
# Initialize stcp object for this test.
stcp <- Stcp$new(method = "ST",
                 family = "Normal",
                 alternative = "greater",
                 threshold = log(1 / 0.05),
                 m_pre = 0)

# Update the observations
obs <- c(1.0, 3.0, 2.0)
stcp$updateLogValuesUntilStop(obs)

# Check whether the sequential test is stopped
stcp$isStopped() # TRUE

# Check when the test was stopped
stcp$getStoppedTime() # 3

# Although the number of obervaions was 4, the test was stopped at 3.
stcp$getTime() # 3

# Get the log value of the mixutre of e-values at the current time (3)
stcp$getLogValue() # 4.425555

# ...which is higher than the threshold log(1 / 0.05) ~ 2.996
stcp$getThreshold() # 2.995732

# Reset the test object
stcp$reset()

# Rerun the test but, at this time, we track updated log values
log_values <- stcp$updateAndReturnHistories(obs)
print(log_values) # 0.1159777 2.7002207 4.4255551 1.9746508
```

```
# Again, the test was stopped at 3rd observation
stcp$getStoppedTime() # 3

# But, at this time, log values were evaluated until the 4th observation.
stcp$getTime() # 4

# Print overall summary
stcp # or stcp$print() or print(stcp)
# stcp Model:
#   - Method:  ST
# - Family:  Normal
# - Alternative:  greater
# - Alpha:  0.05
# - m_pre:  0
# - Num. of mixing components:  55
# - Obs. have been passed:  4
# - Current log value:  1.974651
# - Is stopped before:  TRUE
# - Stopped time:  3
```

Index