Package ‘MSPRT’

August 19, 2019

Type Package

Title A Modified Sequential Probability Ratio Test (MSPRT)

Version 2.1

Date 2019-08-09

Author Sandipan Pramanik [aut, cre],
Valen E. Johnson [aut],
Anirban Bhattacharya [aut]

Maintainer Sandipan Pramanik <sandy@stat.tamu.edu>

Description
A modified SPRT (MSPRT) can be designed and implemented with the help of this package. In a MSPRT design, (i) the maximum sample size of an experiment is fixed prior to the start of an experiment; (ii) the alternative hypothesis used to define the rejection region of the test is derived from the size of the test (Type I error), the maximum available sample size (N), and (iii) the targeted Type 2
error (equals to 1 minus the power) is also prespecified. Given these values, the MSPRT is defined in a
manner very similar to Wald's initial proposal. This test can reduce the average sample size required to
perform statistical hypothesis tests at the specified level of significance and power. This package implements one-sample proportion tests, one-sample Z-tests, one-sample T-tests, two-sample Z-tests and two-sample T-tests. A user guidance for this package is provided here.
One can also refer to the supplemental information for the same.

Imports nleqslv, ggplot2, foreach, iterators, parallel, doParallel,
datasets, graphics, grDevices, methods, stats, utils

NeedsCompilation no

Repository CRAN

Date/Publication 2019-08-19 17:00:03 UTC

R topics documented:

`MSPRT-package` *A Modified Sequential Probability Ratio Test (MSPRT)*

Description

A modified SPRT (MSPRT) can be designed and implemented with the help of this package. In a MSPRT design, (i) the maximum sample size of an experiment is fixed prior to the start of an experiment; (ii) the alternative hypothesis used to define the rejection region of the test is derived from the size of the test (Type I error), the maximum available sample size (N), and (iii) the targeted Type 2 error (equals to 1 minus the power) is also prespecified. Given these values, the MSPRT is defined in a manner very similar to Wald's initial proposal. This test can reduce the average sample size required to perform statistical hypothesis tests at the specified level of significance and power. This package implements one-sample proportion tests, one-sample Z-tests, one-sample T-tests, two-sample Z-tests and two-sample T-tests. A user guidance for this package is provided here. One can also refer to the supplemental information for the same.

Details

Package:	MSPRT
Type:	Package
Version:	2.0
Date:	02-15-2019
License:	GPL>=2

Author(s)

Sandipan Pramanik [aut, cre], Valen E. Johnson [aut], Anirban Bhattacharya [aut]

Maintainer: Sandipan Pramanik <sandy@stat.tamu.edu>

`check` *Sequential checking*

Description

This compares a sequence of values with two sequences of thresholds, namely upper and lower thresholds. At any step, it rejects if the value gets larger than the upper threshold, accepts if it becomes smaller than the lower threshold, and remains inconclusive if it stays in between until the end. If it stays inconclusive at the last step, this function compares the last value with the termination threshold. If the value is larger than the termination threshold, it rejects; otherwise, it accepts.

As a particular case, this function sequentially checks the likelihood ratios with the upper, lower and termination thresholds according to the MSPRT algorithm (Algorithm 1 in the supplemental file).

This function can be readily adapted to Group sequential designs by specifying `batch.seq` (in one-sample tests), and `batch1.seq` & `batch2.seq` (in two-sample tests) accordingly.

Usage

```
check(test.type, statistic, upper, lower,
      batch.seq, batch1.seq, batch2.seq, threshold)
```

Arguments

`test.type`	a character; denotes the type of test; "`oneProp`" for a one-sample proportion test "`oneZ`" for a one-sample Z-test "`oneT`" for a one-sample T-test "`twoZ`" for a two-sample Z-test "`twoT`" for a two-sample T-test
`statistic`	a numeric vector; contains a sequence of values which we want to compare. In particular for the MSPRT, this is a vector of sequentially calculated likelihood ratios (L_n). It's length should not exceed the length of `batch.seq` in one-sample tests, and length of `batch1.seq` (or `batch2.seq`) in two-sample tests.
`upper`	a numeric vector; the sequence of upper threshold. It's length should be equal to the length of `batch.seq` in one-sample tests, and length of `batch1.seq` (or `batch2.seq`) in two-sample tests.
`lower`	a numeric vector; the sequence of lower threshold. It's length should be equal to the length of `batch.seq` in one-sample tests, and length of `batch1.seq` (or `batch2.seq`) in two-sample tests.
`batch.seq`	a numeric vector; **required only in one-sample tests.** an increasing sequence of sample size values where data are supposed to be observed sequentially
`batch1.seq`	a numeric vector; **required only in two-sample tests.** an increasing sequence of sample size values where data from Group-1 are supposed to be observed sequentially
`batch2.seq`	a numeric vector; required only in two-sample tests. an increasing sequence of sample size values where data from Group-2 are supposed to be observed sequentially
`threshold`	a positive numeric; the termination threshold in a MSPRT.

Details

Suppose in a one-sample test, we can affrod at most 100 samples and observe the data after every sample. In this case, the `batch.seq` will be `1:100`.

In another scenario, suppose we observe the data in groups at every 10th sample. In this case, the `batch.seq` will be `seq( from=10,to=100,by=10)`. So there are at most 10 batches/steps where we can observe the data and compare.

Once the L_n's, the upper and lower thresholds, and the termination threshold are provided, this function implements the Algorithm 1 in the supplemental file.

Value

In one-sample tests, this returns a list containing the following elements:

`decision`	a character; the final decision that is made. Either `"accept"`, `"reject"` or `"continue"`.
`n`	a numeric (positive integer); number of samples needed for reaching the `decision`
`exit.stage`	a numeric (positive integer); the step where the decision was reached at

In two-sample tests this returns a similar list as above, except now `n` is replaced by `n1` & `n2` having the following descriptions:

`n1`	a numeric (positive integer); number of samples needed from Group-1 for reaching the `decision`
`n2`	a numeric (positive integer); number of samples needed from Group-2 for reaching the `decision`

Author(s)

Sandipan Pramanik, Valen E. Johnson and Anirban Bhattacharya

`compmerge.list` *Merging two lists componentwise*

Description

Suppose two lists have same number of components. This function creates a new list by merging (or concatenating) them componentwise.

Usage

```
compmerge.list(l1, l2)
```

Arguments

`l1`	a list
`l2`	a list

Details

`l1` and `l2` need to have same number of components.

Value

Returns the merged list.

Author(s)

Sandipan Pramanik, Valen E. Johnson and Anirban Bhattacharya

Examples

```
list1 = list("a"=1, "b"=3, "c"=5)
list2 = list("a"=2, "b"=4, "c"=6)

compmerge.list(list1, list2)
```

`design.MSPRT` *Designing a MSPRT*

Description

Given the desired values of Type 1 & Type 2 error probabilities and the maximum available number of samples (N), this function designs the MSPRT (by finding the 'Termination Threshold' γ). γ is the smallest possible value so that the Type 1 error of the MSPRT is maintained at the desired level. This function designs the MSPRT for one-sample proportion tests, one-sample Z and T-tests, and two-sample Z and T-tests.

By default (that is, if `alt.comp` is missing or `FALSE`) this provides the operating characteristics (OC) for the obtained MSPRT at the null hypothesized value. Otherwise, this also finds the OC at a user desired point alternative. This point alternative is specified through `alt.comp`.

In general, `OC.MSPRT()` can also be used to find the OC of the MSPRT at any desired parameter value.

Usage

```
design.MSPRT(test.type, side, batch.seq, batch1.seq, batch2.seq,
             type1 = 0.005, type2 = 0.2, null, sigma0 = 1,
             N.max, N1.max, N2.max, alt.comp, repl,
             verbose = T, core.no)
```

Arguments

`test.type`	a character; denotes the type of test. “`oneProp`” for a one-sample binomial proportion test. “`oneZ`” for a one-sample Z-test. “`oneT`” for a one-sample T-test. “`twoZ`” for a two-sample Z-test. “`twoT`” for a two-sample T-test.
`side`	a character; direction of the alternative hypothesis H1. Has to be one of `"right"` or `"left"`. **Default:** `"right"`.
`batch.seq`	a numeric vector; **required only in one-sample tests.** Sizes of sequentially observed batches in a group sequential design. **Required only if the design is group sequential, that is when data are observed in batches. Otherwise, only need to provide `N.max`.**
`batch1.seq`	a numeric vector; **required only in two-sample tests.** Sizes of sequentially observed batches from Group-1 in a group sequential design. **Required only if the design is group sequential, that is when data from Group-1 are observed in batches. Otherwise, only need to provide `N1.max`.**
`batch2.seq`	a numeric vector; **required only in two-sample tests.** Sizes of sequentially observed batches from Group-2 in a group sequential design. **Required only if the design is group sequential, that is when data from Group-2 are observed in batches. Otherwise, only need to provide `N2.max`.**
`type1`	a numeric in `(0,1)`; the probability at which we want to control the Type 1 error of the MSPRT. **Default:** `0.005`.
`type2`	a numeric in `(0,1)`; the probability at which we want to control the Type 2 error of the MSPRT. **Default:** `0.2`.
`null`	a numeric; **required only in one-sample proportion and Z-tests.** Denotes value of the hypothesized parameter under the null hypothesis. The hypothesized parameters are proportion in one-sample binomial proportion test, population mean in one-sample Z & T-tests, and difference between the population means of Group-2 and Group-1 in two-sample Z & T-tests. In one-sample T-tests and two-sample tests, only `null=0` is allowed. This is done automatically. This argument is **ignored** in these cases. **Default:** 0.5 in one-sample binomial proportion test, and 0 in one-sample Z-tests.
`sigma0`	a positive numeric; **required only in one & two-sample Z-tests.** Known population standard deviation in one-sample tests and known common population standard deviation in two-sample tests. **Default:** 1.

`N.max`	a positive numeric (integer); **required only in one-sample tests.** Maximum number of samples that we can afford in the one-sample test. In a group sequential design, this should be equal to `sum(batch.seq)`. So in that case, it's enough to provide only `batch.seq` and not `N.max`.
`N1.max`	a positive numeric (integer); **required only in two-sample tests.** Maximum number of samples from Group-1 that we can afford in the two-sample test. In a group sequential design, this should be equal to `sum(batch1.seq)`. So in that case, it's enough to provide only `batch1.seq` and not `N1.max`.
`N2.max`	a positive numeric (integer); **required only in two-sample tests.** Maximum number of samples from Group-2 that we can afford in the two-sample test. In a group sequential design, this should be equal to `sum(batch2.seq)`. So in that case, it's enough to provide only `batch2.seq` and not `N2.max`.
`alt.comp`	missing, `FALSE` or `TRUE`, or a numeric; If missing or `FALSE`, the OC of the MSPRT are obtained only at the `null`; If `TRUE`, the OC of the MSPRT are computed at the `null` and the 'fixed design alternative'; If a numeric, it can be any value under the alternative (consistent with the `"side"`). Then OC of the MSPRT are obtained at the `null` and this point.
`repl`	a positve numeric (integer); total number of replications to be used in Monte Carlo method to calculate the OC of the MSPRT. **Default (Recommended):** `2e+6` in a one-sample proportion test; `1e+6` otherwise. Should be at least `1e+5`.
`verbose`	a logical; if `TRUE`, returns messages of the current proceedings; otherwise it doesn't. **Default:** `TRUE`.
`core.no`	a numeric; number of cores this function can use for carrying out a parallel computation. **Default:** `1` if there are at most 2 cores, otherwise `(number of cores -1)`.

Details

In two-sample tests, the hypothesized parameter is (population mean of Group-2 - population mean of Group-1). So `null=0` implies that these two means are equal under the null hypothesis; `side="right"` implies that the population mean of Group-2 is larger under this alternative hypothesis; `side="left"` implies that the population mean of Group-1 is larger under this alternative hypothesis.

For a user guide, please refer to the supplemental information.

Value

Let us first look at the outputs in one-sample tests.

If `alt.comp` is missing or `FALSE` (default), this computes the OC of the obtained MSPRT under the null hypothesis, and returns a list with the following components:

`type1.est`	a numeric in `(0,1)`; the Type 1 error probability of the MSPRT.

`avg.n0`	a positive numeric; the number of samples required on an average by the MSPRT for coming to a decision when the null hypothesis is true.
`umpbt.alt`	a numeric or a numeric vector of length 2; In one-sample proportion tests, this is usually of length 2. They specify the two points of the UMPBT alternative. In one & two-sample Z-tests, this is the UMPBT point alternative. In one & two-sample T-tests, this is not returned.
`psi.umpbt`	a numeric in `(0,1)`; **returned only in case of one-sample proportion tests.** This denotes the probability of the first component in `umpbt.alt`.
`rej.threshold`	a numeric; the constant value of Wald's rejection threshold.
`acc.threshold`	a numeric; the constant value of Wald's acceptance threshold.
`term.thresh`	a positive numeric; denotes the Termination Threshold (γ) of a MSPRT.

If `alt.comp` equals `TRUE`, this additionally computes the OC of the obtained MSPRT at the 'fixed design alternative'. In this case, the function returns a list with the following components in addition to the previously mentioned components:

`type2.est`	a numeric in `(0,1)`; the Type 2 error probability of the obtained MSPRT at the alternative.
`avg.n1`	a positive numeric; the number of samples required on an average by the MSPRT for coming to a decision when the alternative is true.
`alt`	a numeric; the point alternative where the performance is computed. In this case, this is the 'fixed design alternative'.
`alt.type2`	a numeric in `(0,1)`; the Type 2 error probability of the fixed design test at `alt`. In this case this is exactly `type2`.

If `alt.comp` is numeric, then this computes the list exactly it was for `alt.comp = TRUE` with `type2.est`, `avg.n1`, `alt` and `alt.type2` now computed at this user specified value.

In two-sample tests we get similar outputs as in one-sample tests above, except `avg.n0` and `avg.n1` is replaced by `avg.n1_0`, `avg.n2_0` and `avg.n1_1`, `avg.n2_1`, respectively.

`avg.n1_0`	a positive numeric; the number of samples from Group-1 required on an average by the MSPRT for coming to a decision when the null hypothesis is true.
`avg.n2_0`	a positive numeric; the number of samples from Group-2 required on an average by the MSPRT for coming to a decision when the null hypothesis is true.
`avg.n1_1`	a positive numeric; the number of samples from Group-1 required on an average by the MSPRT for coming to a decision when the alternative hypothesis is true.
`avg.n2_1`	a positive numeric; the number of samples from Group-2 required on an average by the MSPRT for coming to a decision when the alternative hypothesis is true.

Author(s)

Sandipan Pramanik, Valen E. Johnson and Anirban Bhattacharya

References

MSPRT: Main article and Supplemental file

Johnson, Valen E., Uniformly most powerful Bayesian tests., Ann. of Stat., 41, (4), 2013, pp. 1716-1741

Johnson, Valen E., Revised standards for statistical evidence., Proceedings of the National Academy of Sciences, 16, 1945.

Daniel J. Benjamin, James O. Berger, Magnus Johannesson, et al. Redefine statistical significance. Nature Human Behaviour, 2017.

Examples

```
## right-sided one-sample proportion test

# can observe data after each sample sequentially. Max available
# sample size is 30
# design.MSPRT(test.type="oneProp", null = 0.2, N.max = 30)


# can observe data at every fifth sample sequentially. Max available
# sample size is 30. So there are 6 batches each of size 5.
# design.MSPRT(test.type="oneProp", null = 0.2, N.max = 30,
#              batch.seq = rep(5,6))



## right-sided one-sample Z-test

# can observe data after each sample sequentially. Max available
# sample size is 30
# design.MSPRT(test.type="oneZ", null = 3, sigma0 = 1.5,
#              N.max = 30)


# can observe data at every fifth sample sequentially. Max available
# sample size is 30. So there are 6 batches each of size 5.
# design.MSPRT(test.type="oneZ", null = 3, sigma0 = 1.5,
#              N.max = 30, batch.seq = rep(5,6))



## right-sided one-sample T-test

# can observe data after each sample sequentially. Max available
# sample size is 30
# design.MSPRT(test.type="oneT", N.max = 30)


# can observe data at every fifth sample sequentially. Max available
# sample size is 30. So there are 6 batches each of size 5.
# design.MSPRT(test.type="oneT", N.max = 30, batch.seq = rep(5,6))
```

```
## right-sided two-sample Z-test

# can observe data after each sample for each group sequentially.
# Max available sample size is 30 for both groups
# design.MSPRT(test.type="twoZ", sigma0 = 1.5, N1.max = 30, N2.max = 30)


# can observe data at every fifth sample for each group sequentially.
# Max available sample size is 30 for both groups. So there are 6 batches
# each of size 5 for each group.
# design.MSPRT(test.type="twoZ", sigma0 = 1.5,
#              batch1.seq = rep(5,6), batch2.seq = rep(5,6),
#              N1.max = 30, N2.max = 30)



## right-sided two-sample T-test

# can observe data after each sample for each group sequentially.
# Max available sample size is 30 for both groups
# design.MSPRT(test.type="twoT", N1.max = 30, N2.max = 30)


# can observe data at every fifth sample for each group sequentially.
# Max available sample size is 30 for both groups. So there are 6 batches
# each of size 5 for each group.
# design.MSPRT(test.type="twoT", N1.max = 30, N2.max = 30,
#              batch1.seq = rep(5,6), batch2.seq = rep(5,6))
```

`effective.N` *Determining the 'effective maximum sample size' in MSPRT for one-sample proportion tests*

Description

Because of the discreteness issue in one-sample proportion tests, it is not always 'effective' (the sense is explained in the Details section) to use just any value as a maximum sample size for designing a MSPRT. Suppose, we have a desired value for the maximum sample size. Given this, the function finds a value, defined as the effective maximum sample size, which should be used as the maximum sample size for designing the MSPRT (as `N.max` in `design.MSPRT()`).

Usage

```
effective.N(N, side = "right", type1 = 0.005, null = 0.5, plot.it = T)
```

Arguments

`N`	a numeric; a desired value of the maximum sample size for designing the MSPRT.
`side`	a character; direction of the alternative hypothesis H1. Has to be one of `"right"` or `"left"`. **Default:** `"right"`.
`type1`	a numeric in `(0,1)`; the probability at which we want to control the Type 1 error of the MSPRT. **Default:** `0.005`.
`null`	a numeric; the hypothesized value of proportion under the null hypothesis. **Default:** 0.5.
`plot.it`	a logical; if TRUE, returns a plot; otherwise it doesn't. **Default:** TRUE.

Details

Suppose we are provided with a simple null hypothesis, and Type 1 & 2 error probabilities. Because of the discreteness issue in one-sample proportion tests in a fixed design, the fixed design alternative will not always decrease even if we are increasing the sample size. So, we first shortlist only those values from 1 to `N` which results in strictly decreasing UMPBT point alternatives (as is originally defined in Johnson (2013)). The effective maximum sample size is then chosen to be the maximum among those shorlisted value. So, obviously, the effective maximum sample size is either `N` or smaller than that.

Value

Returns a numeric, the effective maximum sample size.

Author(s)

Sandipan Pramanik, Valen E. Johnson and Anirban Bhattacharya

References

Main article and supplemental file of MSPRT

Johnson, Valen E., Uniformly most powerful Bayesian tests., Ann. of Stat., 41, (4), 2013, pp. 1716-1741

Examples

```
effective.N(N = 30, null = .2)
```

error.summary — *Error probabilities of a MSPRT*

Description

This function post processes the outputs of `overshoot.oneProp()`, `overshoot.oneZ()`, `overshoot.oneT()`, `overshoot.twoZ()` and `overshoot.twoT()`, and then calculates the Type 1 or Type 2 error probabilities of a MSPRT.

Usage

```
error.summary(error.type, delta, root, count, inconclusive.vec, R,
              type1, type2)
```

Arguments

`error.type`	a character; “`type1`” for computing Type 1 error probability. “`type2`” for computing Type 2 error probability.
`delta`	a positive numeric; the termination threshold.
`root`	a numeric; given this, (error probability - `root`) is returned.
`count`	a numeric (integer); same as the “`count`” output from `overshoot.oneProp()`, `overshoot.oneZ()`, `overshoot.oneT()`, `overshoot.twoZ()` and `overshoot.twoT()` of the same `error.type`.
`inconclusive.vec`	a numeric vector; same as the “`inconclusive.vec`” output from `overshoot.oneProp()`, `overshoot.oneZ()`, `overshoot.oneT()`, `overshoot.twoZ()` and `overshoot.twoT()` of the same `error.type`.
`R`	a numeric (integer); number of replications in Monte Carlo method that is used in `overshoot.oneProp()`, `overshoot.oneZ()`, `overshoot.oneT()`, `overshoot.twoZ()` and `overshoot.twoT()` of the same `error.type`.
`type1`	a numeric in `(0,1)`; the probability at which we want to control the Type 1 error of the MSPRT.
`type2`	a numeric in `(0,1)`; the probability at which we want to control the Type 2 error of the MSPRT.

Details

Suppose we have the Monte Carlo outputs from `overshoot.oneProp()` or `overshoot.oneZ()` or `overshoot.oneT()` or `overshoot.twoZ()` or `overshoot.twoT()` corresponding to some `error.type`. Given these, the function returns the difference (error probability - `root`) for the MSPRT whose termination threshold is `delta`.

Value

a numeric; the difference (error probability of the MSPRT - `root`).

Author(s)

Sandipan Pramanik, Valen E. Johnson and Anirban Bhattacharya

`find.alt` *Finding the 'fixed design alternative'*

Description

This function obtains the point alternative in one-sample binomial proportion tests, one & two-sample Z-tests, and one & two-sample T-tests in a fixed design.

Usage

```
find.alt(test.type, side = "right", null, n, n1, n2,
         type1 = 0.005, type2 = 0.2, sigma0 = 1)
```

Arguments

`test.type`	a character; denotes the type of test; "`oneProp`" for a one-sample binomial proportion test. "`oneZ`" for a one-sample Z-test. "`oneT`" for a one-sample T-test. "`twoZ`" for a two-sample Z-test. "`twoT`" for a two-sample T-test.
`side`	a character; direction of the alternative hypothesis H1. Has to be one of `"right"` or `"left"`. **Default:** `"right"`.
`null`	a numeric; **required only in one-sample proportion tests and one-sample Z-tests**; hypothesized value of the parameter under the null hypothesis. The hypothesized parameters are proportion in one-sample proportion tests, population mean in one-sample Z & T-tests, and difference between the population mean of Group-2 and Group-1 in two-sample Z & T-tests. In one-sample T-tests and two-sample Z & T-tests, only `null=0` is allowed. This is done automatically. This argument is **ignored** in these cases. **Default:** 0.5 in one-sample proportion tests, and 0 in one-sample Z-tests.
`n`	a positive numeric (integer); **required only in one-sample tests**; number of samples on which the fixed design one-sample test is based on.
`n1`	a positive numeric (integer); **required only in two-sample tests**; number of samples from Group-1 on which the fixed design two-sample test is based on.

`n2`	a positive numeric (integer); **required only in two-sample tests**; number of samples from Group-2 on which the fixed design two-sample test is based on.
`type1`	a numeric in `(0,1)`; Type 1 error probability of the test. **Default:** `0.005`.
`type2`	a numeric in `(0,1)`; Type 2 error probability. Corresponding to this, the point alternative is obtained. **Default:** `0.2`.
`sigma0`	a positive numeric; **required only in one & two-sample Z-tests.** The known population standard deviation in one-sample Z-tests. The known common population standard deviation in two-sample Z-tests. **Default:** 1.

Details

At the 'fixed design alternative', the fixed design test of size `type1` has Type 2 error `type2`.

In one-sample tests, the fixed design is based on `n` samples. In two-sample tests, the fixed design is based on `n1` & `n2` samples from Group 1 & 2, respectively.

Value

Returns a numeric which is the obtained 'fixed design alternative'.

Author(s)

Sandipan Pramanik, Valen E. Johnson and Anirban Bhattacharya

Examples

```
## finding the alternative in case of a one-sample proportion test
## which provides 80% power against a right and left sided
## alternative, respectively

# default null = 0.5
find.alt(test.type="oneProp", n= 60, type1= 0.005, type2= 0.2)

find.alt(test.type="oneProp", side= "left", n= 60,
         type1= 0.005, type2= 0.2)

# null = 0.2
find.alt(test.type="oneProp", null= 0.2, n= 60,
         type1= 0.005, type2= 0.2)

find.alt(test.type="oneProp", side= "left", null= 0.2,
         n= 60, type1= 0.005, type2= 0.2)


## finding the alternative in case of a one-sample Z-test
## which provides 80% power against a right and left sided
## alternative, respectively
```

```
# default sigma0 = 1

# default null = 0
find.alt(test.type="oneZ", n= 60, type1= 0.005, type2= 0.2)

find.alt(test.type="oneZ", side= "left", n= 60, type1= 0.005,
         type2= 0.2)

# null = 3
find.alt(test.type="oneZ", null= 3, n= 60, type1= 0.005, type2= 0.2)

find.alt(test.type="oneZ", side= "left", null= 3, n= 60,
         type1= 0.005, type2= 0.2)


## finding the alternative in case of a one-sample T-test
## which provides 80% power against a right and left sided
## alternative, respectively

# default null = 0
find.alt(test.type="oneT", n= 60, type1= 0.005, type2= 0.2)

find.alt(test.type="oneT", side= "left", n= 60,
         type1= 0.005, type2= 0.2)

# null = 3
find.alt(test.type="oneT", null= 3, n= 60, type1= 0.005,
         type2= 0.2)

find.alt(test.type="oneT", side= "left", null= 3,
         n= 60, type1= 0.005, type2= 0.2)


## finding the alternative in case of a two-sample Z-test
## which provides 80% power against a right and left sided
## alternative, respectively
# default sigma0 = 1

find.alt(test.type="twoZ", n1= 60, n2= 50, type1= 0.005, type2= 0.2)

find.alt(test.type="twoZ", side= "left", n1= 60, n2= 50, type1= 0.005,
         type2= 0.2)


## finding the alternative in case of a two-sample T-test
## which provides 80% power against a right and left sided
## alternative, respectively

find.alt(test.type="twoT", n1= 60, n2= 50, type1= 0.005, type2= 0.2)

find.alt(test.type="twoT", side= "left", n1= 60, n2= 50, type1= 0.005,
         type2= 0.2)
```

find.samplesize — *Sample size required to achieve a higher significance for one-sample tests in a fixed design*

Description

This function finds the sample size that is required to maintain a desired power at a point alternative when we decrease the level of significance of a fixed design one-sample test. This can be calculated in one-sample proportion tests, and one-sample Z & T-tests.

Usage

```
find.samplesize(test.type, N, lower.signif = 0.05, higher.signif = 0.005,
                null, side = "right", pow = 0.8, alt, sigma0 = 1,
                n.seq, verbose=T, plot.it = T)
```

Arguments

`test.type`	a character; denotes the type of test; “`oneProp`” for a one-sample proportion tests. “`oneZ`” for a one-sample Z-tests. “`oneT`” for a one-sample T-tests.
`N`	a positive numeric (integer); **required only in one-sample tests**; number of samples on which the fixed design one-sample test is based on.
`lower.signif`	a numeric in `(0,1)`; denotes the lower level of significance. **Default:** `0.05`.
`higher.signif`	a numeric in `(0,1)`; denotes the higher level of significance. **Default:** `0.005`.
`null`	a numeric; denotes the hypothesized value of the parameter under the null hypothesis. The hypothesized parameters are proportion in one-sample proportion tests, and population mean in one-sample Z & T-tests. **Default:** 0.5 in one-sample proportion tests, and 0 in one-sample Z and T-tests.
`side`	a character; direction of the alternative hypothesis H1. Has to be one of `"right"` or `"left"`. **Default:** `"right"`.
`pow`	a numeric in `(0,1)`; desired level of power at the point alternative `alt`. **Default:** 0.8. This means 80 percent power.
`alt`	missing or a numeric; value of the point alternative where we want to maintain the power `pow`. **Default:** the ‘fixed design alternative’ at the `lower.signif` using `N` samples.

`sigma0`	a positive numeric; **required only in one-sample Z-tests.** known population standard deviation. **Default:** 1.
`n.seq`	missing or a numeric vector; the final value of increased sample size is searched over this vector. **Default:** `N:(4*N)`.
`verbose`	logical; if `TRUE`, returns a message of the achieved results; otherwise it doesn't. **Default:** `TRUE`.
`plot.it`	a logical; if `TRUE`, returns a plot; otherwise it doesn't. **Default:** `TRUE`.

Value

Returns a numeric. In a fixed design for the specified one-sample test, this is the sample size that we require to achieve `higher.signif` while still mainting at least pow amount of power at `alt`.

Author(s)

Sandipan Pramanik, Valen E. Johnson and Anirban Bhattacharya

References

Main article and supplemental file of MSPRT

Examples

```
find.samplesize(test.type = "oneProp", N = 30, null = .2)

# In this case, the fixed design alternative at 0.05 is 0.4263. As it seems,
# we need to increase the sample size to 51 to achieve the higher significance
# of 0.005 while still maintaining at least 80% power at 0.4263.


find.samplesize(test.type = "oneProp", N = 30, null = .2, side = "left")

# In this case, the fixed design alternative at 0.05 is 0.0516. For testing
# against the left sided alternative, we need to increase the sample size to
# 66 to achieve the higher significance of 0.005 while still maintaining at
# least 80% power at 0.0516.

find.samplesize(test.type = "oneZ", N = 30)
find.samplesize(test.type = "oneZ", N = 30, side = "left")

find.samplesize(test.type = "oneT", N = 30)
find.samplesize(test.type = "oneT", N = 30, side = "left")
```

find.threshold.oneProp

Optimizing the UMPBT objective function in fixed design one-sample proportion tests

Description

Given δ, this function finds the difference (`optimum value of the objective function -a constant`) in one-sample proportion tests in a fixed design. Notation is as in the supplemental information.

Usage

```
find.threshold.oneProp(delta, side = "right", n, p0 = 0.5, opt.interval, root)
```

Arguments

`delta`	a positive numeric; corresponding to this, the UMPBT alternative (UMPBT(δ)) is obtained.
`side`	a character; direction of the alternative hypothesis H1. Has to be one of `"right"` or `"left"`. **Default:** `"right"`.
`n`	a positive numeric (integer); number of samples on which the fixed design one-sample test is based on.
`p0`	a numeric in `(0,1)`; the hypothesized value of proportion under the simple null hypothesis. **Default:** 0.5.
`opt.interval`	a numeric vector of length 2; contains the lower and upper endpoints of an interval to be optimized over. Endpoints should lie inside `(0,1)`. **Default:** `c(p0,1)` if `side = "right"`, and `c(0,p0)` if `side = "left"`.
`root`	a numeric; the 'constant' in the above description. Given this, the function retunrs (`optimum value of the objective function -root`).

Details

Apart from finding the optimum value of the objective function, the argument `root` can be used to solve for a `delta`. For example, using `root=k` we can find a `delta` such that the optimized value of the objective function is `k`.

Value

If `root=k`, this returns a numeric denoting the difference (`optimum value of the objective function -k`).

Author(s)

Sandipan Pramanik, Valen E. Johnson and Anirban Bhattacharya

Examples

```
## returns minimum value of the objective function
find.threshold.oneProp(delta= 25, n= 60, p0= 0.2, root= 0)

## returns (minimum value of the objective function -5)
find.threshold.oneProp(delta= 25, n= 60, p0= 0.2, root= 5)
```

`implement.MSPRT` *Implementing a MSPRT*

Description

Once we have designed the MSPRT (that is, obtained the termination threshold, γ) using `design.MSPRT()`, this function implements the MSPRT algorithm for a given data. This is done by sequentially calculating the likelihood ratio(s) or bayes factor(s) (L_n), and then comparing them with the acceptance and rejection thresholds.

This function implements the MSPRT in one-sample proportion tests, and one & two-sample Z & T-tests.

Usage

```
implement.MSPRT(test.type, obs, obs1, obs2, side,
                batch.seq, batch1.seq, batch2.seq,
                type1 = 0.005, type2 = 0.2, null, sigma0, term.thresh,
                N.max, N1.max, N2.max, plot.it = T, verbose = T)
```

Arguments

`test.type`	a character; denotes the type of test. “`oneProp`” for a one-sample binomial proportion test. “`oneZ`” for a one-sample Z-test. “`oneT`” for a one-sample T-test. “`twoZ`” for a two-sample Z-test. “`twoT`” for a two-sample T-test.
`obs`	a numeric vector; **required only in one-sample tests.** Denotes the sequentially observed data based on which we want to carry out the null hypothesis significance testing (NHST) using a one-sample test.
`obs1`	a numeric vector; **required only in two-sample tests.** Denotes the sequentially observed data from Group-1 based on which we want to carry out the NHST using a two-sample test.

`obs2`	a numeric vector; **required only in two-sample tests.** Denotes the sequentially observed data from Group-2 based on which we want to carry out the NHST using a two-sample test.
`side`	a character; direction of the alternative hypothesis H1. Has to be one of `"right"` or `"left"`. **Default:** `"right"`.
`batch.seq`	a numeric vector; **required only in one-sample tests.** Sizes of sequentially observed batches in a group sequential design. **Required only if the design is group sequential, that is when data are observed in batches. Otherwise, only need to provide `N.max`.**
`batch1.seq`	a numeric vector; **required only in two-sample tests.** Sizes of sequentially observed batches from Group-1 in a group sequential design. **Required only if the design is group sequential, that is when data from Group-1 are observed in batches. Otherwise, only need to provide `N1.max`.**
`batch2.seq`	a numeric vector; **required only in two-sample tests.** Sizes of sequentially observed batches from Group-2 in a group sequential design. **Required only if the design is group sequential, that is when data from Group-2 are observed in batches. Otherwise, only need to provide `N2.max`.**
`type1`	a numeric in `(0,1)`; the probability at which we want to control the Type 1 error of the MSPRT. **Default:** `0.005`.
`type2`	a numeric in `(0,1)`; the probability at which we want to control the Type 2 error of the MSPRT. **Default:** `0.2`.
`null`	a numeric; **required only in one-sample tests.** the hypothesized value of parameter under the null hypothesis. The hypothesized parameters are proportion in one-sample proportion tests, population mean in one & two-sample Z & T-tests, and difference between the population mean of Group-2 and Group-1 in two-sample Z & T-tests. In two-sample tests, only `null=0` is allowed. This is done automatically. This argument is **ignored** in these cases. **Default:** 0.5 in one-sample binomial proportion test, and 0 in one-sample Z and T-tests.
`sigma0`	a positive numeric; **required only in one & two-sample Z-tests.** Known population standard deviation in one-sample tests and known common population standard deviation in two-sample tests. **Default:** 1.
`term.thresh`	a positive numeric; denotes the termination threshold of a MSPRT. This is determined at the designing step of a MSPRT using `design.MSPRT()`.
`N.max`	a positive numeric (integer); **required only in one-sample tests.** Maximum number of samples that we can afford in the one-sample test. In a group sequential design, this should be equal to `sum(batch.seq)`. So in that case, it's enough to provide only `batch.seq` and not `N.max`.

`N1.max`	a positive numeric (integer); **required only in two-sample tests.** Maximum number of samples from Group-1 that we can afford in the two-sample test. In a group sequential design, this should be equal to `sum(batch1.seq)`. So in that case, it's enough to provide only `batch1.seq` and not `N1.max`.
`N2.max`	a positive numeric (integer); **required only in two-sample tests.** Maximum number of samples from Group-2 that we can afford in the two-sample test. In a group sequential design, this should be equal to `sum(batch2.seq)`. So in that case, it's enough to provide only `batch2.seq` and not `N2.max`.
`plot.it`	logical vector; if `TRUE` (**Default**), a comparison plot is returned. Otherwise it's not.
`verbose`	a logical; if `TRUE`, returns messages of the current proceedings; otherwise it doesn't. **Default:** `TRUE`.

Details

Suppose we want to carry out one of the above tests. To do that, we simply need to follow two steps:

Step 1: Designing the MSPRT: Determine the 'Termination threshold' using `design.MSPRT()`.

Step 2: Implementing the MSPRT: Implement the MSPRT algorithm for a sequentially observed data using `implement.MSPRT()`. In Step 2, we need to use the termination threshold obtained from Step 1.

Value

Returns a list with the following components:

`decision`	a character; the decision reached based on the provided data. Either `"reject"` (means 'Reject H0') or `"accept"` (means 'Don't reject H0') or `"continue"` (means 'continue sampling').
`n`	a numeric (integer); number of samples required for reaching the `decision`.
`lhood.ratio`	a numeric vector; a vector of likelihood ratios (L_n 's) in favor of the UMPBT alternative. This is computed at each element of `batch.seq` until the available data or the maximum available sample size.
`rej.threshold`	a numeric; Wald's rejection threshold.
`acc.threshold`	a numeric; Wald's acceptance threshold.
`umpbt.alt`	a numeric or numeric vector; denotes the UMPBT alternative(s). At each stage, likelihood ratios (L_n 's) are computed in favor of this alternative.
`psi.umpbt`	a numeric; denotes the mixing probability for the UMPBT alternative in one-sample proportion tests. **Returned only in one-sample proportion tests.**

Author(s)

Sandipan Pramanik, Valen E. Johnson and Anirban Bhattacharya

References

MSPRT: Main article and Supplemental file

Johnson, Valen E., Uniformly most powerful Bayesian tests., Ann. of Stat., 41, (4), 2013, pp. 1716-1741

Johnson, Valen E., Revised standards for statistical evidence., Proceedings of the National Academy of Sciences, 16, 1945.

Daniel J. Benjamin, James O. Berger, Magnus Johannesson, et al. Redefine statistical significance. Nature Human Behaviour, 2017.

Examples

```
# the termination thresholds are obtained from design.MSPRT()

## one-sample tests

### proportion test

x = c(0, 0, 0, 0, 1, 0, 1, 1, 1, 0, 1, 0, 0, 1, 0,
      0, 1, 0, 1, 1, 1, 1, 0, 0, 0, 1, 0, 1, 0, 0)

implement.MSPRT(obs = x, test.type = "oneProp", null = 0.2,
                term.thresh = 22.63, N.max = 30)


### Z-test

x = c(4.060319, 5.275465, 3.746557, 7.392921, 5.494262,
      3.769297, 5.731144, 6.107487, 5.863672)

implement.MSPRT(obs = x, test.type = "oneZ", null = 3,
                sigma0 = 1.5, term.thresh = 27.856, N.max = 30)


### T-test

x = c(1.738717, 5.076539, 1.116762, 3.105214, 5.567161, 2.095638,
      2.291750, 2.046943, 2.571340, 3.207162, 4.841446, 1.797331)

implement.MSPRT(obs = x, test.type = "oneT", null = 3,
                term.thresh = 33.152, N.max = 30)



## two-sample tests

### Z-test

x1 = c(0.6546282, 2.2772738, 4.3817680, 0.3044365, 1.8796224,
      2.1986304, 3.0619321, 1.6404530, 4.9767109)   # group-1
```

```
x2 = c(0.5570999, 1.5612114, 2.3881823, 0.2718022, 2.2936742,
       2.0451859, 2.1281266, 3.6749153, 0.1717139)   # group-2

implement.MSPRT(test.type = "twoZ", obs1 = x1, obs2 = x2,
                sigma0 = 1.5, term.thresh = 27.928,
                N1.max = 30, N2.max = 30)


### T-test

x1 = c(-0.93968072,  0.27546499, -1.25344292,  2.39292120,  0.49426166,
       -1.23070258,  0.73114358,  1.10748706,  0.86367203, -0.45808258,
        2.26767175,  0.58476485, -0.93186087, -3.32204983,  1.68739638,
       -0.06740041, -0.02428539,  1.41575432,  1.23183179)   # group-1

x2 = c( 0.6546282,  2.2772738,  4.3817680,  0.3044365,  1.8796224,
        2.1986304,  3.0619321,  1.6404530,  4.9767109,  1.7918195,
        2.6264761,  3.4726292,  1.4109570,  0.4404965,  4.6733434,
       -1.4666036,  3.3179069,  2.0537101,  3.5192430)   # group-2

implement.MSPRT(test.type = "twoT", obs1 = x1, obs2 = x2,
                term.thresh = 32.972, N1.max = 30, N2.max = 30)
```

LR.oneProp *Likelihood ratio in one-sample proportion tests*

Description

Given a simple null and a simple alternative hypotheses, this function calculates the likelihood ratio (LR) in favor of the alternative based on an observed data in one-sample proportion tests.

Usage

```
LR.oneProp(m, suff.stat, null = 0.5, alt)
```

Arguments

`m`	a postive numeric (integer); number of samples for computing the LR.
`suff.stat`	a postive numeric (integer); the value of sufficient statistic based on `m` observed data. In this case, the sufficient statistic is the total no. of successes out of `m` observations.
`null`	a numeric in `(0,1)`; the hypothesized value of proportion under the simple null. **Default:** 0.5.
`alt`	a numeric in `(0,1)`; the hypothesized value of proportion under the simple alternative.

Value

Returns a numeric denoting the LR in favor of `alt` in the one-sample proportion test based on `m` observations.

Author(s)

Sandipan Pramanik, Valen E. Johnson and Anirban Bhattacharya

References

MSPRT: supplemental information

Examples

```
LR.oneProp(m= 60, suff.stat= 48, null= 0.2, alt= 0.5)
```

`LR.oneT` *Bayes factor in one-sample T-tests*

Description

Given a simple null and a simple alternative hypotheses, this function calculates the bayes factor (BF) in favor of the alternative based on an observed data in one-sample T-tests.

Usage

```
LR.oneT(m, suff.stat, null = 0, alt, s)
```

Arguments

`m`	a postive numeric (integer); number of samples for computing the BF.
`suff.stat`	numeric; the value of sufficient statistic based on `m` observed data. In this case, the sufficient statistic is the sum of `m` observations.
`null`	a numeric; the hypothesized value of population mean under the simple null. **Default:** 0.
`alt`	a numeric; the hypothesized value of population mean under the simple alternative.
`s`	a positive numeric; sample standard deviation (with divisor `(m-1)`) of the `m` observations.

Value

Returns a numeric denoting the BF in favor of `alt` in one-sample T-tests based on `m` observations.

Author(s)

Sandipan Pramanik, Valen E. Johnson and Anirban Bhattacharya

References

MSPRT: supplemental information

Examples

```
LR.oneT(m= 60, suff.stat= 20.2, alt= 1.5, s= 1.2)
```

LR.oneZ *Likelihood ratio in one-sample Z-tests*

Description

Given a simple null and a simple alternative hypotheses, this function calculates the likelihood ratio (LR) in favor of the alternative based on an observed data in one-sample Z-tests.

Usage

```
LR.oneZ(m, suff.stat, null = 0, alt, sigma0 = 1)
```

Arguments

`m`	a postive numeric (integer); number of samples for computing the LR.
`suff.stat`	numeric; the value of sufficient statistic based on `m` observed data. In this case, the sufficient statistic is the sum of `m` observations
`null`	a numeric; the hypothesized value of population mean under the simple null. **Default:** 0.
`alt`	a numeric; the hypothesized value of population mean under the simple alternative.
`sigma0`	a positive numeric; the known population standard deviation (sd). **Default:** 1.

Value

Returns a numeric denoting the LR in favor of `alt` in the one-sample Z-test based on `m` observations.

Author(s)

Sandipan Pramanik, Valen E. Johnson and Anirban Bhattacharya

References

MSPRT: supplemental information

Examples

```
LR.oneZ(m= 60, suff.stat= 10.5, alt= 0.5)
```

LR.twoT *Bayes factor in two-sample T-tests*

Description

Given the simple null equals to 0 and a simple alternative hypothesis, this function calculates the bayes factor (BF) in favor of the alternative based on an observed data in two-sample T-tests.

Usage

```
LR.twoT(m1, m2, suff.stat1, suff.stat2, alt, s)
```

Arguments

`m1`	a postive numeric (integer); number of samples from Group-1 for computing the BF.
`m2`	a postive numeric (integer); number of samples from Group-2 for computing the BF.
`suff.stat1`	numeric; the value of sufficient statistic based on `m1` samples from Group-1. In this case, the sufficient statistic is the mean of these `m1` observations.
`suff.stat2`	numeric; the value of sufficient statistic based on `m2` samples from Group-2. In this case, the sufficient statistic is the mean of these `m2` observations.
`alt`	a numeric; the hypothesized value under the simple alternative.
`s`	a positive numeric; pooled sample standard deviation using `m1` observations from Group-1 and `m2` observations from Group-2.

Value

Returns a numeric denoting the BF in favor of `alt` in the two-sample T-test based on the observations.

Author(s)

Sandipan Pramanik, Valen E. Johnson and Anirban Bhattacharya

References

MSPRT: supplemental information

Examples

```
LR.twoT(m1= 60, m2= 50, suff.stat1= 20, suff.stat2= 19, alt= 1, s= 1)
```

`LR.twoZ` *Likelihood ratio in two-sample Z-tests*

Description

Given the simple null equals to 0 and a simple alternative hypotheses, this function calculates the likelihood ratio (LR) in favor of the alternative based on an observed data in two-sample Z-tests.

Usage

```
LR.twoZ(m1, m2, suff.stat1, suff.stat2, alt, sigma0 = 1)
```

Arguments

`m1`	a postive numeric (integer); number of samples from Group-1 for computing the LR.
`m2`	a postive numeric (integer); number of samples from Group-2 for computing the LR.
`suff.stat1`	numeric; the value of sufficient statistic based on `m1` samples from Group-1. In this case, the sufficient statistic is the mean of those `m1` observations.
`suff.stat2`	numeric; the value of sufficient statistic based on `m2` samples from Group-2. In this case, the sufficient statistic is the mean of those `m2` observations.
`alt`	a numeric; hypothesized value under the simple alternative
`sigma0`	a positive numeric; the known common population standard deviation (sd) **Default** is 1.

Value

Returns a numeric denoting the LR in favor of `alt` in the two-sample Z-test based on the observations.

Author(s)

Sandipan Pramanik, Valen E. Johnson and Anirban Bhattacharya

References

MSPRT: supplemental information

Examples

```
LR.twoZ(m1= 60, m2= 50, suff.stat1= 20, suff.stat2= 19, alt= 1, sigma0= 1.2)
```

`objfunc.oneProp`	*Objective function for determining the UMPBT point alternative in one-sample proportion tests*

Description

This is the $h(p, \delta)$ function as in the supplemental file. Given a δ, we optimize this function to get the UMPBT(δ) alternative for one-sample proportion tests in a fixed design.

Usage

```
objfunc.oneProp(p, delta, n, p0)
```

Arguments

`p`	a numeric in `(0,1)`; the value of proportion.
`delta`	a positive numeric; corresponding to this δ, the UMPBT(δ) alternative is obtained.
`n`	a positive numeric (integer); number of samples to be used.
`p0`	a numeric in `(0,1)`; the hypothesized value of proportion under the simple null.

Value

Returns a numeric which is the value of the objective function.

Author(s)

Sandipan Pramanik, Valen E. Johnson and Anirban Bhattacharya

References

MSPRT: Supplemental information

Johnson, Valen E., Uniformly most powerful Bayesian tests., Ann. of Stat., 41, (4), 2013, pp. 1716-1741

Examples

```
objfunc.oneProp(p= .5, delta= 25, n= 60, p0= 0.2)
```

OC.MSPRT *Operating characteristics for a MSPRT*

Description

This function evaluates the operating characteristics (OC) for a MSPRT at any specified value of the hypothesized parameter. If the specified value lies in the region of the alternative hypothesis, this computes the Type 2 error probability of the MSPRT; otherwise this computes the Type 1 error probability. In both cases, this also computes the number of samples those are required on an average for making a decision.

Usage

```
OC.MSPRT(test.type, side, batch.seq, batch1.seq, batch2.seq,
         null, term.thresh, theta, sigma0, type1 = 0.005, type2 = 0.2,
         N.max, N1.max, N2.max, verbose = T, repl, core.no)
```

Arguments

`test.type`	a character; denotes the type of test. “`oneProp`” for a one-sample binomial proportion test. “`oneZ`” for a one-sample Z-test. “`oneT`” for a one-sample T-test. “`twoZ`” for a two-sample Z-test. “`twoT`” for a two-sample T-test.
`side`	a character; direction of the alternative hypothesis H1. Has to be one of `"right"` or `"left"`. **Default:** `"right"`.
`batch.seq`	a numeric vector; **required only in one-sample tests.** Sizes of sequentially observed batches in a group sequential design. **Required only if the design is group sequential, that is when data are observed in batches. Otherwise, only need to provide `N.max`.**
`batch1.seq`	a numeric vector; **required only in two-sample tests.** Sizes of sequentially observed batches from Group-1 in a group sequential design. **Required only if the design is group sequential, that is when data from Group-1 are observed in batches. Otherwise, only need to provide `N1.max`.**
`batch2.seq`	a numeric vector; **required only in two-sample tests.** Sizes of sequentially observed batches from Group-2 in a group sequential design. **Required only if the design is group sequential, that is when data from Group-2 are observed in batches. Otherwise, only need to provide `N2.max`.**
`null`	a numeric; **required only in one-sample tests.** Denotes value of the hypothesized parameter under the null hypothesis.

	The hypothesized parameters are proportion in one-sample binomial proportion test, population mean in one-sample Z & T-tests, and difference between the population means of Group-2 and Group-1 in two-sample Z & T-tests. In two-sample tests, only `null=0` is allowed. This is done automatically. This argument is **ignored** in these cases. **Default:** 0.5 in one-sample binomial proportion test, and 0 in one-sample Z-tests.
`term.thresh`	a positive numeric; denotes the termination threshold of a MSPRT. This is determined at the designing step of a MSPRT using `design.MSPRT()`.
`theta`	a numeric; the value of hypothesized parameter where we want to evaluate the OC for the MSPRT.
`sigma0`	a positive numeric; **required only in one & two-sample Z-tests.** Known population standard deviation in one-sample tests and known common population standard deviation in two-sample tests. **Default:** 1.
`type1`	a numeric in `(0,1)`; the probability at which we want to control the Type 1 error of the MSPRT. **Default:** `0.005`.
`type2`	a numeric in `(0,1)`; the probability at which we want to control the Type 2 error of the MSPRT. **Default:** `0.2`.
`N.max`	a positive numeric (integer); **required only in one-sample tests.** Maximum number of samples that we can afford in the one-sample test. In a group sequential design, this should be equal to `sum(batch.seq)`. So in that case, it's enough to provide only `batch.seq` and not `N.max`.
`N1.max`	a positive numeric (integer); **required only in two-sample tests.** Maximum number of samples from Group-1 that we can afford in the two-sample test. In a group sequential design, this should be equal to `sum(batch1.seq)`. So in that case, it's enough to provide only `batch1.seq` and not `N1.max`.
`N2.max`	a positive numeric (integer); **required only in two-sample tests.** Maximum number of samples from Group-2 that we can afford in the two-sample test. In a group sequential design, this should be equal to `sum(batch2.seq)`. So in that case, it's enough to provide only `batch2.seq` and not `N2.max`.
`verbose`	a logical; if `TRUE`, returns messages of the current proceedings; otherwise it doesn't. **Default:** `TRUE`.
`repl`	a positve numeric (integer); total number of replications to be used in Monte Carlo method to calculate the OC of the MSPRT. **Default (Recommended):** `2e+6` in a one-sample proportion test; `1e+6` otherwise. Should be at least `1e+5`.
`core.no`	a numeric; number of cores this function can use for carrying out a parallel computation. **Default:** `1` if there are at most 2 cores, otherwise `(number of cores -1)`.

Details

For `side="right"`, if `theta>null` then the Type 2 error is calculated, where as the Type 1 error is calculated when `theta<=null`; and vice versa.

To put it simply, if `theta` falls under the alternative hypothesis, then Type 2 error and `avg.n1` (in one-sample tests) or `avg.n1_1` & `avg.n2_1` (in two-sample tests) are calculated; otherwise Type 1 error and `avg.n0` (in one-sample tests) or `avg.n1_0` & `avg.n2_0` (in two-sample tests) are calculated.

Value

Returns a list with the following components:

In one-sample tests:

`type1.est` or `type2.est`
: a numeric in `(0,1)`; the Type 1 or Type 2 error probability of the MSPRT evaluated at `theta`, respectively.

`avg.n0` or `avg.n1`
: a positive numeric; the number of samples required on an average by the MSPRT for reaching a decision when `theta` is actually true.

`n0.vec` or `n1.vec`
: a numeric vector; denotes a vector of sample sizes required for reaching a decision in each of the `repl` replications in the Monte-Carlo study when `theta` is actually true.
: This is a vector of length `repl` .

In two-sample tests:

`type1.est` or `type2.est`
: a numeric in `(0,1)`; the Type 1 or Type 2 error probability of the MSPRT evaluated at `theta`, respectively.

`avg.n1_0` & `avg.n2_0`, or `avg.n1_1` & `avg.n2_1`
: a positive numeric; the number of samples required on an average by the MSPRT from Group 1 and 2 for reaching a decision when `theta` is actually true.

`n1_0.vec` & `n2_0.vec`, or `n1_1.vec` & `n2_1.vec`
: a numeric vector; denotes a vector of sample sizes required from Group 1 and 2 for reaching a decision in each of the `repl` replications in the Monte-Carlo study when `theta` is actually true.
: This is a vector of length `repl` .

Author(s)

Sandipan Pramanik, Valen E. Johnson and Anirban Bhattacharya

Examples

```
# the termination thresholds are obtained from design.MSPRT()
```

```
## One-sample proportion test
## finding OC at theta = 0.3

# OC.MSPRT(test.type = "oneProp", null = 0.2,
#          term.thresh = 22.63, theta = 0.3, N.max = 30)



## One-sample Z-test
## finding OC at theta = 4

# OC.MSPRT(test.type = "oneZ", null = 3, sigma0 = 1.5,
#          term.thresh = 27.856, theta = 4, N.max = 30)


## One-sample T-test
## finding OC at theta = 4

# OC.MSPRT(test.type = "oneT", null = 3,
#          term.thresh = 33.152, theta = 4, N.max = 30)


## Two-sample Z-test
## finding OC at theta = 1

# OC.MSPRT(test.type = "twoZ", sigma0 = 1.5, term.thresh = 27.928,
#          theta = 1, N1.max = 30, N2.max = 30)


## Two-sample T-test
## finding OC at theta  = 1

# OC.MSPRT(test.type = "twoT", term.thresh = 32.972,
#          theta = 1, N1.max = 30, N2.max = 30)
```

overshoot.oneProp — *Error summary of the truncated Wald's SPRT in one-sample proportion tests*

Description

MSPRT is designed on the assumption that we can afford at most, say, `N` samples. This function calculates a summary of the Type 1 or Type 2 error committed by the Wald's SPRT when it is simply truncated at `N` in one-sample proportion tests. This is done using parallel computation.

It is worth a mention that a case may remain inconclusive due to the truncation. The required sample size for reaching that decision is `N`.

Usage

```
overshoot.oneProp(error.type, batch.seq, null, gen.par, alt.LR, alt.psi,
                  up, low, N, R, core.no, return.n = T)
```

Arguments

`error.type`	a character; specifies which of the 2 types of errors need to be accounted for. `"type1"` for Type 1 error. `"type2"` for Type 2 error.
`batch.seq`	a numeric vector; an increasing sequence of values until `N`. Denotes the sequence of sample sizes where a user will observe data sequentially. Last element should equal to `N`.
`null`	a numeric in `(0,1)`; the hypothesized value of proportion under the simple null hypothesis.
`gen.par`	a numeric; the value of proportion from which the data needs to be generated from.
`alt.LR`	a numeric vector of lenth 2; this consists of the 2 UMPBT alternative points. The sequence of weighted likelihood ratios (L_n) is computed in favour of this alternative. This is same with the output `u` from `umpbt.oneProp()`.
`alt.psi`	a numeric in `(0,1)`; the mixing probability corresponding to the first component of `alt.LR` in the UMPBT alternative. This is same with the output `psi` from `umpbt.oneProp()`.
`up`	a numeric; value of a constant rejection threshold. Should be greater than `low`.
`low`	a numeric; value of a constant acceptance threshold. Should be smaller than `up`.
`N`	a positive numeric (integer); number of samples where truncation of Wald's SPRT is required. In a MSPRT, this is the maximum available sample size.
`R`	a positive numeric (integer); number of replications desired in the Monte Carlo study. At least `1e+5` is required.
`core.no`	a numeric; number of cores this function can use for carrying out the Monte Carlo study using the parallel computing.
`return.n`	logical; if `TRUE`, this returns a vector of sample sizes required for reaching a decision in each of the `R` replications.

Value

If `return.n = TRUE`, a list with following components is returned:

`count`	a numeric; number of errors of `error.type` those are committed out of `R` replications.
`inconclusive.vec`	a numeric vector; a vector containing the likelihood ratio values at `N` which remained inconclusive at the last stage.

`n.vec`	a numeric vector; a vector of the required number of samples in each replications. This vector has length `R`.

If `return.n = FALSE`, the same list except `n.vec` is returned.

Author(s)

Sandipan Pramanik, Valen E. Johnson and Anirban Bhattacharya

References

Wald, A., Sequential Tests of Statistical Hypotheses. Ann. of Math. Statist., vol. 16, no. 2, 1945, 117-186.

Examples

```
N.max = 30
#overshoot.oneProp( error.type= "type1", batch.seq= 1:N.max, null= 0.5,
#                   gen.par= 0.5, alt.LR= c(0.5,0.55), alt.psi= 0.4,
#                   up= 160, low= 0.2, N= N.max, R= 1e+6,
#                   core.no= 2, return.n = T)

#overshoot.oneProp( error.type= "type2", batch.seq= 1:N.max, null= 0.5,
#                   gen.par= 0.7, alt.LR= c(0.5,0.55), alt.psi= 0.4,
#                   up= 160, low= 0.2, N= N.max, R= 1e+6,
#                   core.no= 2, return.n = T)
```

`overshoot.oneT` *Error summary of the truncated Wald's SPRT in one-sample T-tests*

Description

MSPRT is designed on the assumption that we can afford at most, say, `N` samples. This function calculates a summary of the Type 1 or Type 2 error committed by the Wald's SPRT when it is simply truncated at `N` in one-sample T-tests. This is done using parallel computation.

It is worth a mention that a case may remain inconclusive due to the truncation. The required sample size for reaching that decision is `N`.

Usage

```
overshoot.oneT(side, error.type, batch.seq, type1, null, gen.par,
                up, low, N, R, core.no, return.n = T)
```

Arguments

`side`	a character; direction of the alternative hypothesis H1. Has to be one of `"right"` or `"left"`.
`error.type`	a character; specifies which of the 2 types of errors need to be accounted for. `"type1"` for Type 1 error. `"type2"` for Type 2 error.
`batch.seq`	a numeric vector; an increasing sequence of values until `N`. Denotes the sequence of sample sizes where a user will observe data sequentially. First element should be at least `2`. Last element should equal to `N`.
`type1`	a numeric in `(0,1)`; the probability at which we want to control the Type 1 error of the MSPRT.
`null`	a numeric; the hypothesized value of population mean under the simple null hypothesis.
`gen.par`	a numeric; the value of population mean from which normal observations need to be generated from.
`up`	a numeric; value of a constant rejection threshold. Should be greater than `low`.
`low`	a numeric; value of a constant acceptance threshold. Should be smaller than `up`.
`N`	a positive numeric (integer); number of samples where truncation of Wald's SPRT is required. In a MSPRT, this is the maximum available sample size.
`R`	a positive numeric (integer); number of replications desired in the Monte Carlo study. At least `1e+5` is required.
`core.no`	a numeric; number of cores this function can use for carrying out the Monte Carlo study using the parallel computing.
`return.n`	logical; if `TRUE`, this returns a vector of sample sizes required for reaching a decision in each of the `R` replications.

Value

If `return.n = TRUE`, a list with following components is returned:

`count`	a numeric; number of errors of `error.type` those are committed out of `R` replications.
`inconclusive.vec`	a numeric vector; a vector containing the likelihood ratio values at `N` which remained inconclusive at the last stage.
`n.vec`	a numeric vector; a vector of the required number of samples in each replications. This vector has length `R`.

If `return.n = FALSE`, the same list except `n.vec` is returned.

Author(s)

Sandipan Pramanik, Valen E. Johnson and Anirban Bhattacharya

References

Wald, A., Sequential Tests of Statistical Hypotheses. Ann. of Math. Statist., vol. 16, no. 2, 1945, 117-186.

Examples

```
N.max = 30
#overshoot.oneT( side="right", error.type= "type1", batch.seq= 2:N.max, null= 0,
#             type1 = 0.005, gen.par= 0, up= 160, low= 0.2, N= N.max, R= 1e+6, core.no= 2,
#                return.n = T)

#overshoot.oneT( side="right", error.type= "type2", batch.seq= 2:N.max, null= 0,
#            type1 = 0.005, gen.par= 1.5, up= 160, low= 0.2, N= N.max, R= 1e+6, core.no= 2,
#                return.n = T)
```

`overshoot.oneZ` *Error summary of the truncated Wald's SPRT in one-sample Z-tests*

Description

MSPRT is designed on the assumption that we can afford at most, say, `N` samples. This function calculates a summary of the Type 1 or Type 2 error committed by the Wald's SPRT when it is simply truncated at `N` in one-sample Z-tests. This is done using parallel computation.

It is worth a mention that a case may remain inconclusive due to the truncation. The required sample size for reaching that decision is `N`.

Usage

```
overshoot.oneZ(error.type, batch.seq, null, gen.par, alt.LR,
               up, low, N, R, core.no, return.n = T)
```

Arguments

`error.type`	a character; specifies which of the 2 types of errors need to be accounted for. `"type1"` for Type 1 error. `"type2"` for Type 2 error.
`batch.seq`	a numeric vector; an increasing sequence of values until `N`. Denotes the sequence of sample sizes where a user will observe data sequentially. Last element should equal to `N`.
`null`	a numeric; the hypothesized value of population mean under the simple null hypothesis.
`gen.par`	a numeric vector of length 2; the first component is the value of the population mean and second component is the known standard deviation (sd). Observations are generated from a normal distribution with this mean and sd.

`alt.LR`	a numeric; the simple alternative in favor of which the likelihood ratios (L_n) are calculated sequentially. The UMPBT point alternative is used in case of a MSPRT. This is same with the output `u` from `umpbt.oneZ()`.
`up`	a numeric; value of a constant rejection threshold; should be greater than `low`.
`low`	a numeric; value of a constant acceptance threshold; should be smaller than `up`.
`N`	a positive numeric (integer); number of samples where truncation of Wald's SPRT is required; in a MSPRT, this is the maximum available sample size.
`R`	a positive numeric (integer); number of replications desired in the Monte Carlo study; at least `1e+5` is required
`core.no`	a numeric; number of cores this function can use for carrying out the Monte Carlo study using the parallel computing.
`return.n`	logical; if `TRUE`, this returns a vector of sample sizes required for reaching a decision in each of the `R` replications.

Value

If `return.n = TRUE`, a list with following components is returned:

`count`	a numeric; number of errors of `error.type` those are committed out of `R` replications
`inconclusive.vec`	a numeric vector; a vector containing the values of L_N which remained inconclusive at the last stage
`n.vec`	a numeric vector; a vector of the required number of samples; this is of length `R`.

If `return.n = FALSE`, the same list except `n.vec` is returned.

Author(s)

Sandipan Pramanik, Valen E. Johnson and Anirban Bhattacharya

References

Wald, A., Sequential Tests of Statistical Hypotheses. Ann. of Math. Statist., vol. 16, no. 2, 1945, 117-186.

Examples

```
N.max = 30
#overshoot.oneZ( error.type= "type1", batch.seq= 1:N.max, null= 0,
#                gen.par= c(0,1), alt.LR= 1, up= 160, low= 0.2, N= N.max,
#                R= 1e+6, core.no= 2, return.n = T)
```

```
#overshoot.oneZ( error.type= "type2", batch.seq= 1:N.max, null= 0,
#                gen.par= c(1.5,1), alt.LR= 1, up= 160, low= 0.2, N= N.max,
#                R= 1e+6, core.no= 2, return.n = T)
```

`overshoot.twoT` *Error summary of the truncated Wald's SPRT in two-sample T-tests*

Description

MSPRT is designed on the assumption that we can afford at most, say, `N` samples. This function calculates a summary of the Type 1 or Type 2 error committed by the Wald's SPRT when it is simply truncated at `N` in two-sample T-tests. This is done using parallel computation.

It is worth a mention that a case may remain inconclusive due to the truncation. The required sample size for reaching that decision is `N`.

Usage

```
overshoot.twoT(side, error.type, batch1.seq, batch2.seq, type1, gen.par,
               up, low, N1, N2, R, core.no, return.n = T)
```

Arguments

`side`	a character; direction of the alternative hypothesis H1. Has to be one of `"right"` or `"left"`.
`error.type`	a character; specifies which of the 2 types of errors need to be accounted for. `"type1"` for Type 1 error. `"type2"` for Type 2 error.
`batch1.seq`	a numeric vector; an increasing sequence of values until `N1`. Denotes the sequence of sample sizes where data is observed sequentially from Group-1. First element should be at least `2`. Last element should equal be to `N1`.
`batch2.seq`	a numeric vector; an increasing sequence of values until `N2`. Denotes the sequence of sample sizes where data is observed sequentially from Group-2. First element should be at least `2`. Last element should equal be to `N2`.
`type1`	a numeric in `(0,1)`; the probability at which we want to control the Type 1 error of the MSPRT.
`gen.par`	a numeric; observations from Group-1 and 2 are generated from the normal distributions with common variance 1, and means `-gen.par` and `gen.par`, respectively.
`up`	a numeric; value of a constant rejection threshold. Should be greater than `low`.
`low`	a numeric; value of a constant acceptance threshold. Should be smaller than `up`.
`N1`	a positive numeric (integer); maximum available number of samples from Group-1.

`N2`	a positive numeric (integer); maximum available number of samples from Group-2.
`R`	a positive numeric (integer); number of replications desired in the Monte Carlo study. At least `1e+5` is required.
`core.no`	a numeric; number of cores this function can use for carrying out the Monte Carlo study using the parallel computing.
`return.n`	logical; if `TRUE`, this returns a vector of sample sizes required for reaching a decision in each of the `R` replications.

Value

If `return.n = TRUE`, a list with following components is returned:

`count`	a numeric; number of errors of `error.type` those are committed out of `R` replications.
`inconclusive.vec`	a numeric vector; a vector containing the likelihood ratio values at `N` which remained inconclusive at the last stage.
`n.vec`	a numeric vector; a vector of the required number of samples in each replications. This vector has length `R`.

If `return.n = FALSE`, the same list except `n.vec` is returned.

Author(s)

Sandipan Pramanik, Valen E. Johnson and Anirban Bhattacharya

References

Wald, A., Sequential Tests of Statistical Hypotheses. Ann. of Math. Statist., vol. 16, no. 2, 1945, 117-186.

Examples

```
N1.max = 30
N2.max = 30
#overshoot.twoT( side="right", error.type= "type1",
#                batch1.seq= 2:N1.max,  batch2.seq= 2:N2.max,
#                type1= 0.005, gen.par= 0, up= 160, low= 0.2,
#                N1= N1.max, N2= N2.max, R= 1e+6, core.no= 2, return.n = T)

#overshoot.twoT( side="right", error.type= "type2",
#                batch1.seq= 2:N1.max,  batch2.seq= 2:N2.max,
#                type1= 0.005, gen.par= 1, up= 160, low= 0.2,
#                N1= N1.max, N2= N2.max, R= 1e+6, core.no= 2, return.n = T)
```

overshoot.twoZ *Error summary of the truncated Wald's SPRT in two-sample Z-tests*

Description

MSPRT is designed on the assumption that we can afford at most, say, `N` samples. This function calculates a summary of the Type 1 or Type 2 error committed by the Wald's SPRT when it is simply truncated at `N` in two-sample Z-tests. This is done using parallel computation.

It is worth a mention that a case may remain inconclusive due to the truncation. The required sample size for reaching that decision is `N`.

Usage

```
overshoot.twoZ(error.type, batch1.seq, batch2.seq, gen.par,
               alt.LR, up, low, N1, N2, R, core.no, return.n = T)
```

Arguments

`error.type`	a character; specifies which of the 2 types of errors need to be accounted for. `"type1"` for Type 1 error. `"type2"` for Type 2 error.
`batch1.seq`	a numeric vector; an increasing sequence of values until `N1`. Denotes the sequence of sample sizes where data is observed sequentially from Group-1. Last element should equal be to `N1`.
`batch2.seq`	a numeric vector; an increasing sequence of values until `N2`. Denotes the sequence of sample sizes where data is observed sequentially from Group-2. Last element should equal be to `N2`.
`gen.par`	a numeric; observations from Group-1 and 2 are generated from the normal distributions with common standard deviation `gen.par[2]`, and means `-gen.par[1]` and `gen.par[1]`, respectively.
`alt.LR`	a numeric; the simple alternative in favor of which the likelihood ratios are calculated sequentially. The UMPBT point alternative is used in case of a MSPRT. This is same with the output `u` from `umpbt.twoZ()`.
`up`	a numeric; value of a constant rejection threshold. Should be greater than `low`.
`low`	a numeric; value of a constant acceptance threshold. Should be smaller than `up`.
`N1`	a positive numeric (integer); maximum available number of samples from Group-1.
`N2`	a positive numeric (integer); maximum available number of samples from Group-2.
`R`	a positive numeric (integer); number of replications desired in the Monte Carlo study. At least `1e+5` is required.

`core.no`	a numeric; number of cores this function can use for carrying out the Monte Carlo study using the parallel computing.
`return.n`	logical; if `TRUE`, this returns a vector of sample sizes required for reaching a decision in each of the `R` replications.

Value

If `return.n = TRUE`, a list with following components is returned:

`count`	a numeric; number of errors of `error.type` those are committed out of `R` replications.
`inconclusive.vec`	a numeric vector; a vector containing the likelihood ratio values at `N` which remained inconclusive at the last stage.
`n.vec`	a numeric vector; a vector of the required number of samples in each replications. This vector has length `R`.

If `return.n = FALSE`, the same list except `n.vec` is returned.

Author(s)

Sandipan Pramanik, Valen E. Johnson and Anirban Bhattacharya

References

Wald, A., Sequential Tests of Statistical Hypotheses. Ann. of Math. Statist., vol. 16, no. 2, 1945, 117-186.

Examples

```
N1.max = 30
N2.max = 30
#overshoot.twoZ( error.type= "type1", batch1.seq= 1:N1.max,  batch2.seq= 1:N2.max,
#                gen.par= c(0,1), alt.LR = 1, up= 160, low= 0.2,
#                N1= N1.max, N2= N2.max, R= 1e+6, core.no= 2, return.n = T)

#overshoot.twoZ( error.type= "type1", batch1.seq= 1:N1.max,  batch2.seq= 1:N2.max,
#                gen.par= c(1,1), alt.LR = 1, up= 160, low= 0.2,
#                N1= N1.max, N2= N2.max, R= 1e+6, core.no= 2, return.n = T)
```

`ovr.repl.oneProp` *A particular replication step in* `overshoot.oneProp()`

Description

In one-sample proportion tests this function simulates a data, computes the weighted likelihood ratios, and compares with the acceptance and rejection thresholds. `overshoot.oneProp()` carries out a Monte Carlo method by repeating this function for `R` number of times.

Usage

```
ovr.repl.oneProp(error.type, batch.seq, null, gen.par, alt.LR, alt.psi,
                 up, low, N, seed)
```

Arguments

`error.type`	a character; specifies which of the 2 types of errors need to be accounted for. `"type1"` for Type 1 error. `"type2"` for Type 2 error.
`batch.seq`	a numeric vector; an increasing sequence of values until `N`. Denotes the sequence of sample sizes where a user will observe data sequentially. Last element should equal to `N`.
`null`	a numeric in `(0,1)`; the hypothesized value of proportion under the simple null hypothesis.
`gen.par`	a numeric; the value of proportion from which the data needs to be generated from.
`alt.LR`	a numeric vector of lenth 2; this consists of the 2 UMPBT alternative points. The sequence of weighted likelihood ratios (L_n) is computed in favour of this alternative. This is same with the output `u` from `umpbt.oneProp()`.
`alt.psi`	a numeric in `(0,1)`; the mixing probability corresponding to the first component of `alt.LR` in the UMPBT alternative. This is same with the output `psi` from `umpbt.oneProp()`.
`up`	a numeric; value of a constant rejection threshold. Should be greater than `low`.
`low`	a numeric; value of a constant acceptance threshold. Should be smaller than `up`.
`N`	a positive numeric (integer); number of samples where truncation of Wald's SPRT is required. In a MSPRT, this is the maximum available sample size.
`seed`	a positive integer; used in `set.seed()` to recreate the simulated data.

Value

Returns a list with following components:

`incr.count`	either 0 or 1; 1 if and only if an error of `error.type` is made.
`inconclusive`	a numeric; the value of L_N if and only if it remains inconclusive after truncating Wald's SPRT at `N`; otherwise a numeric of length 0 is returned.
`n`	a numeric; number of samples required for reaching the decision. In an inconclusive case, this value is `N`.

Author(s)

Sandipan Pramanik, Valen E. Johnson and Anirban Bhattacharya

References

Wald, A., Sequential Tests of Statistical Hypotheses. Ann. of Math. Statist., vol. 16, no. 2, 1945, 117-186.

Examples

```
N.max = 30
ovr.repl.oneProp( error.type= "type1", batch.seq= 1:N.max, null= 0.5, gen.par= 0.5,
                  alt.LR= c(0.5,0.55), alt.psi= 0.4, up= 160, low= 0.2, N= N.max,
                  seed= 1)

ovr.repl.oneProp( error.type= "type2", batch.seq= 1:N.max, null= 0.5, gen.par= 0.7,
                  alt.LR= c(0.5,0.55), alt.psi= 0.4, up= 160, low= 0.2, N= N.max,
                  seed= 1)
```

`ovr.repl.oneT` *A particular replication step in* `overshoot.oneT()`

Description

In one-sample T-tests this function simulates a data, computes the bayes factors, and compares with the acceptance and rejection thresholds. `overshoot.oneT()` carries out a Monte Carlo method by repeating this function for `R` number of times.

Usage

```
ovr.repl.oneT(side, error.type, batch.seq, type1, null, gen.par,
              up, low, N, seed)
```

Arguments

`side`	a character; direction of the alternative hypothesis `H1`. Has to be one of `"right"` or `"left"`.
`error.type`	a character; specifies which of the 2 types of errors need to be accounted for. `"type1"` for Type 1 error. `"type2"` for Type 2 error.
`batch.seq`	a numeric vector; an increasing sequence of values until `N`. Denotes the sequence of sample sizes where a user will observe data sequentially. First element should be at least `2`. Last element should equal to `N`.
`type1`	a numeric in `(0,1)`; the probability at which we want to control the Type 1 error of the MSPRT.
`null`	a numeric; the hypothesized value of population mean under the simple null hypothesis.
`gen.par`	a numeric; the value of population mean from which normal observations need to be generated from.
`up`	a numeric; value of a constant rejection threshold. Should be greater than `low`.
`low`	a numeric; value of a constant acceptance threshold. Should be smaller than `up`.
`N`	a positive numeric (integer); number of samples where truncation of Wald's SPRT is required. In a MSPRT, this is the maximum available sample size.
`seed`	a positive integer; used in `set.seed()` to recreate the simulated data.

Value

Returns a list with following components:

`incr.count`	either 0 or 1; 1 if and only if an error of `error.type` is made.
`inconclusive`	a numeric; the value of L_N if and only if it remains inconclusive after truncating Wald's SPRT at `N`; otherwise a numeric of length 0 is returned.
`n`	a numeric; number of samples required for reaching the decision. In an inconclusive case, this value is `N`.

Author(s)

Sandipan Pramanik, Valen E. Johnson and Anirban Bhattacharya

References

Wald, A., Sequential Tests of Statistical Hypotheses. Ann. of Math. Statist., vol. 16, no. 2, 1945, 117-186.

Examples

```
N.max = 30
ovr.repl.oneT( side = "right", error.type= "type1", batch.seq= 2:N.max,
               type1 = 0.005, null= 0, gen.par= 0, up= 160, low= 0.2,
               N= N.max, seed= 1)

ovr.repl.oneT( side = "right", error.type= "type2", batch.seq= 2:N.max,
               type1 = 0.005, null= 0, gen.par= 1.5, up= 160, low= 0.2,
               N= N.max, seed= 1)
```

`ovr.repl.oneZ` *A particular replication step in* `overshoot.oneZ()`

Description

In one-sample Z-tests this function simulates a data, computes the likelihood ratios, and compares with the acceptance and rejection thresholds. `overshoot.oneZ()` carries out a Monte Carlo method by repeating this function for `R` number of times.

Usage

```
ovr.repl.oneZ(error.type, batch.seq, null, gen.par, alt.LR, up, low, N, seed)
```

Arguments

`error.type`	a character; specifies which of the 2 types of errors need to be accounted for. `"type1"` for Type 1 error. `"type2"` for Type 2 error.
`batch.seq`	a numeric vector; an increasing sequence of values until `N`. Denotes the sequence of sample sizes where a user will observe data sequentially. Last element should equal to `N`.
`null`	a numeric; the hypothesized value of population mean under the simple null hypothesis.
`gen.par`	a numeric vector of length 2; the first component is the value of the population mean and second component is the known standard deviation (sd). Observations are generated from a normal distribution with this mean and sd.
`alt.LR`	a numeric; the simple alternative in favor of which the likelihood ratios (L_n) are calculated sequentially. The UMPBT point alternative is used in case of a MSPRT. This is same with the output `u` from `umpbt.oneZ()`.
`up`	a numeric; value of a constant rejection threshold; should be greater than `low`.
`low`	a numeric; value of a constant acceptance threshold; should be smaller than `up`.

`N`	a positive numeric (integer); number of samples where truncation of Wald's SPRT is required; in a MSPRT, this is the maximum available sample size.
`seed`	a positive integer; used in `set.seed()` to recreate the simulated data.

Value

Returns a list with following components:

`incr.count`	either 0 or 1; 1 if and only if an error of `error.type` is made.
`inconclusive`	a numeric; the value of L_N if and only if it remains inconclusive after truncating Wald's SPRT at `N`; otherwise a numeric of length 0 is returned.
`n`	a numeric; number of samples required for reaching the decision. In an inconclusive case, this value is `N`.

Author(s)

Sandipan Pramanik, Valen E. Johnson and Anirban Bhattacharya

References

Wald, A., Sequential Tests of Statistical Hypotheses. Ann. of Math. Statist., vol. 16, no. 2, 1945, 117-186.

Examples

```
N.max = 30
ovr.repl.oneZ( error.type= "type1", batch.seq= 1:N.max, null= 0,
               gen.par= c(0,1), alt.LR= 1, up= 160, low= 0.2,
               N= N.max, seed= 1)

ovr.repl.oneZ( error.type= "type2", batch.seq= 1:N.max, null= 0,
               gen.par= c(1.5,1), alt.LR= 1, up= 160, low= 0.2,
               N= N.max, seed= 1)
```

`ovr.repl.twoT` *A particular replication step in* `overshoot.twoT()`

Description

In two-sample T-tests this function simulates a data, computes the bayes factors, and compares with the acceptance and rejection thresholds. `overshoot.twoT()` carries out a Monte Carlo method by repeating this function for `R` number of times.

Usage

```
ovr.repl.twoT(side, error.type, batch1.seq, batch2.seq, type1, gen.par,
              up, low, N1, N2, seed)
```

Arguments

`side`	a character; direction of the alternative hypothesis H1. Has to be one of `"right"` or `"left"`.
`error.type`	a character; specifies which of the 2 types of errors need to be accounted for. `"type1"` for Type 1 error. `"type2"` for Type 2 error.
`batch1.seq`	a numeric vector; an increasing sequence of values until `N1`. Denotes the sequence of sample sizes where data is observed sequentially from Group-1. First element should be at least `2`. Last element should equal be to `N1`.
`batch2.seq`	a numeric vector; an increasing sequence of values until `N2`. Denotes the sequence of sample sizes where data is observed sequentially from Group-2. First element should be at least `2`. Last element should equal be to `N2`.
`type1`	a numeric in `(0,1)`; the probability at which we want to control the Type 1 error of the MSPRT.
`gen.par`	a numeric; observations from Group-1 and 2 are generated from the normal distributions with common variance 1, and means `-gen.par` and `gen.par`, respectively.
`up`	a numeric; value of a constant rejection threshold. Should be greater than `low`.
`low`	a numeric; value of a constant acceptance threshold. Should be smaller than `up`.
`N1`	a positive numeric (integer); maximum available number of samples from Group-1.
`N2`	a positive numeric (integer); maximum available number of samples from Group-2.
`seed`	a positive integer; used in `set.seed()` to recreate the simulated data.

Value

Returns a list with following components:

`incr.count`	either 0 or 1; 1 if and only if an error of `error.type` is made.
`inconclusive`	a numeric; the value of L_N if and only if it remains inconclusive after truncating Wald's SPRT at `N`; otherwise a numeric of length 0 is returned.
`n`	a numeric; number of samples required for reaching the decision. In an inconclusive case, this value is `N`.

Author(s)

Sandipan Pramanik, Valen E. Johnson and Anirban Bhattacharya

References

Wald, A., Sequential Tests of Statistical Hypotheses. Ann. of Math. Statist., vol. 16, no. 2, 1945, 117-186.

Examples

```
N1.max = 30
N2.max = 30
ovr.repl.twoT( side="right", error.type= "type1",
               batch1.seq= 2:N1.max,  batch2.seq= 2:N2.max,
               type1= 0.005, gen.par= 0, up= 160, low= 0.2,
               N1= N1.max, N2= N2.max, seed= 1)

ovr.repl.twoT( side="right", error.type= "type2",
               batch1.seq= 2:N1.max,  batch2.seq= 2:N2.max,
               type1= 0.005, gen.par= 1, up= 160, low= 0.2,
               N1= N1.max, N2= N2.max, seed= 1)
```

`ovr.repl.twoZ` *A particular replication step in* `overshoot.twoZ()`

Description

In two-sample Z-tests this function simulates a data, computes the likelihood ratios, and compares with the acceptance and rejection thresholds. `overshoot.twoZ()` carries out a Monte Carlo method by repeating this function for `R` number of times.

Usage

```
ovr.repl.twoZ(error.type, batch1.seq, batch2.seq, alt.LR, gen.par,
              up, low, N1, N2, seed)
```

Arguments

`error.type`	a character; specifies which of the 2 types of errors need to be accounted for. `"type1"` for Type 1 error. `"type2"` for Type 2 error.
`batch1.seq`	a numeric vector; an increasing sequence of values until `N1`. Denotes the sequence of sample sizes where data is observed sequentially from Group-1. Last element should equal be to `N1`.
`batch2.seq`	a numeric vector; an increasing sequence of values until `N2`. Denotes the sequence of sample sizes where data is observed sequentially from Group-2. Last element should equal be to `N2`.

`gen.par`	a numeric; observations from Group-1 and 2 are generated from the normal distributions with common standard deviation `gen.par[2]`, and means `-gen.par[1]` and `gen.par[1]`, respectively.
`alt.LR`	a numeric; the simple alternative in favor of which the likelihood ratios are calculated sequentially. The UMPBT point alternative is used in case of a MSPRT. This is same with the output `u` from `umpbt.twoZ()`.
`up`	a numeric; value of a constant rejection threshold. Should be greater than `low`.
`low`	a numeric; value of a constant acceptance threshold. Should be smaller than `up`.
`N1`	a positive numeric (integer); maximum available number of samples from Group-1.
`N2`	a positive numeric (integer); maximum available number of samples from Group-2.
`seed`	a positive integer; used in `set.seed()` to recreate the simulated data.

Value

Returns a list with following components:

`incr.count`	either 0 or 1; 1 if and only if an error of `error.type` is made.
`inconclusive`	a numeric; the value of L_N if and only if it remains inconclusive after truncating Wald's SPRT at `N`; otherwise a numeric of length 0 is returned.
`n`	a numeric; number of samples required for reaching the decision. In an inconclusive case, this value is `N`.

Author(s)

Sandipan Pramanik, Valen E. Johnson and Anirban Bhattacharya

References

Wald, A., Sequential Tests of Statistical Hypotheses. Ann. of Math. Statist., vol. 16, no. 2, 1945, 117-186.

Examples

```
N1.max = 30
N2.max = 30

## common sd 1

# both groups have mean 0
ovr.repl.twoZ( error.type= "type1", batch1.seq= 1:N1.max,  batch2.seq= 1:N2.max,
               gen.par= c(0,1), alt.LR = 1, up= 160, low= 0.2,
               N1= N1.max, N2= N2.max, seed= 1)

# Group-1 & 2 have mean -1 and 1, respectively
```

```
ovr.repl.twoZ( error.type= "type2", batch1.seq= 1:N1.max,  batch2.seq= 1:N2.max,
               gen.par= c(1,1), alt.LR = 1, up= 160, low= 0.2,
               N1= N1.max, N2= N2.max, seed= 1)
```

point.umpbt.oneProp *The UMPBT point alternative in one-sample proportion tests*

Description

This function finds the UMPBT point alternative, as defined in Johnson (2013), in case of one-sample proportion tests. This is obtained by matching the rejection region of the UMPBT to that of the UMP (or fixed design) test.

Usage

```
point.umpbt.oneProp(side = "right", type1 = 0.005, n, null = 0.5)
```

Arguments

`side`	a character; direction of the alternative hypothesis H1. Has to be one of `"right"` or `"left"`. **Default:** `"right"`.
`type1`	a numeric in `(0,1)`; the prespecified Type 1 error probability. **Default:** `0.005`.
`n`	a positive integer; number of samples to be used.
`null`	a numeric in `(0,1)`; the hypothesized value of proportion under the simple null hypothesis. **Default** is `0.5`.

Value

Returns a numeric which is the UMPBT point alternative as defined in Johnson (2013).

Author(s)

Sandipan Pramanik, Valen E. Johnson and Anirban Bhattacharya

References

Johnson, Valen E., Uniformly most powerful Bayesian tests., Ann. of Stat., 41, (4), 2013, pp. 1716-1741

Examples

```
point.umpbt.oneProp(n= 60, null= .2)
point.umpbt.oneProp(side= "left", n= 60, null= .2)
```

type2.error.oneProp *Type 2 error function of one-sample proportion tests in a fixed design*

Description

For one-sample proportion tests in a fixed design, this function evaluates (Type 2 error - a constant) at a specified alternative value of proportion.

Usage

```
type2.error.oneProp(alt, side = "right", null = 0.5, n, type1 = 0.005, root = 0)
```

Arguments

`alt`	a numeric in `(0,1)`; the value of proportion (consistent with `side`) where the Type 2 error of the fixed design test needs to be evaluated at.
`side`	a character; direction of the alternative hypothesis H1. Has to be one of `"right"` or `"left"`. **Default:** `"right"`.
`null`	a numeric in `(0,1)`; the hypothesized value of proportion under a simple null hypothesis. **Default:** `0.5`.
`n`	a positive numeric (integer); sample size to be used.
`type1`	a numeric in `(0,1)`; prespecified Type 1 error probability. **Default:** `0.005`.
`root`	a numeric; when this is 0, the Type 2 error is returned. In general, `(Type 2 error -root)` is returned. **Default:** `0`.

Details

In case of one-sample proportion tests in a fixed design, this function evaluates the Type 2 error at a specified value `alt`. We can also use this function to do the other way round by exploiting the argument `root`; that is, given a Type 2 error β, we can obtain the `alt` where Type 2 error equals β. To do that, we need to substitute `root=`β in the argument of this function, and then solve it for `alt`. The function `find.alt()` in this package excatly does this.

Value

If `root=k`, a numeric `(Type 2 error -k)` evaluated at the speified alternative value `alt` is returned.

Author(s)

Sandipan Pramanik, Valen E. Johnson and Anirban Bhattacharya

Examples

```
## Type 2 error at an alternative value
type2.error.oneProp(alt= 0.5, null= 0.2, n= 60)

## (Type 2 error - 0.5) at the same alternative value
type2.error.oneProp(alt= 0.5, null= 0.2, n= 60, root = 0.5)
```

`type2.error.oneT` *Type 2 error function of one-sample T-tests in a fixed design*

Description

For a one-sample T-test in a fixed design, this function evaluates (Type 2 error - a constant) at a specified alternative value of population mean.

Usage

```
type2.error.oneT(alt, side = "right", null = 0, n, type1 = 0.005, root = 0)
```

Arguments

`alt`	a numeric; the value of population mean (consistent with `side`) where the Type 2 error of the fixed design test needs to be evaluated at.
`side`	a character; direction of the alternative hypothesis H1. Has to be one of `"right"` or `"left"`. **Default:** `"right"`.
`null`	a numeric; the hypothesized value of population mean under a simple null hypothesis. **Default:** `0`.
`n`	a positive numeric (integer); sample size to be used.
`type1`	a numeric in `(0,1)`; prespecified Type 1 error probability. **Default:** `0.005`.
`root`	a numeric; when this is 0, the Type 2 error is returned; in general, `(Type 2 error -root)` is returned **Default:** `0`.

Details

In case of one-sample T-tests in a fixed design, this function evaluates the Type 2 error at a specified value `alt`. We can also use this function to do the other way round by exploiting the argument `root`; that is, given a Type 2 error β, we can obtain the `alt` where Type 2 error equals β. To do that, we need to substitute `root=`β in the argument of this function, and then solve it for `alt`. The function `find.alt()` in this package excatly does this.

Value

If `root=k`, a numeric `(Type 2 error -k)` evaluated at the speified alternative value `alt` is returned.

Author(s)

Sandipan Pramanik, Valen E. Johnson and Anirban Bhattacharya

Examples

```
## Type 2 error at an alternative value
type2.error.oneT(alt= 1.2, n= 60)

## (Type 2 error - 0.5) at the same alternative value
type2.error.oneT(alt= 1.2, n= 60, root = 0.5)
```

`type2.error.oneZ` *Type 2 error function of one-sample Z-tests in a fixed design*

Description

For one-sample Z-tests in a fixed design, this function evaluates (Type 2 error - a constant) at a specified alternative value of population mean.

Usage

```
type2.error.oneZ(alt, side = "right", null = 0, sigma0 = 1,
                 n, type1 = 0.005, root = 0)
```

Arguments

`alt`	a numeric; the value of population mean (consistent with `side`) where the Type 2 error of the fixed design test needs to be evaluated at.
`side`	a character; direction of the alternative hypothesis H1. Has to be one of `"right"` or `"left"`. **Default:** `"right"`.
`null`	a numeric; the hypothesized value of population mean under a simple null hypothesis. **Default:** `0`.
`sigma0`	a positive numeric; the known population standard deviation. **Default:** 1.
`n`	a positive numeric (integer); sample size to be used.
`type1`	a numeric in `(0,1)`; prespecified Type 1 error probability. **Default:** `0.005`.
`root`	a numeric; when this is 0, the Type 2 error is returned; in general, `(Type 2 error -root)` is returned **Default:** `0`.

Details

In case of one-sample Z-tests in a fixed design, this function evaluates the Type 2 error at a specified value `alt`. We can also use this function to do the other way round by exploiting the argument `root`; that is, given a Type 2 error β, we can obtain the `alt` where Type 2 error equals β. To do that, we need to substitute `root=`β in the argument of this function, and then solve it for `alt`. The function `find.alt()` in this package excatly does this.

Value

If `root=k`, a numeric `(Type 2 error -k)` evaluated at the speified alternative value `alt` is returned.

Author(s)

Sandipan Pramanik, Valen E. Johnson and Anirban Bhattacharya

Examples

```
## Type 2 error at an alternative value
type2.error.oneZ(alt=1.2, n= 60)

## (Type 2 error - 0.5) at the same alternative value
type2.error.oneZ(alt=1.2, n= 60, root = 0.5)
```

`type2.error.twoT` *Type 2 error function of two-sample T-tests in a fixed design*

Description

For a two-sample T-test in a fixed design, this function evaluates (Type 2 error - a constant) at a specified alternative value of the hypothesized parameter (the difference between the population means of Group-2 and Group-1). The hypothesized value under the simple null hypothesis is set at 0.

Usage

```
type2.error.twoT(alt, side = "right", n1, n2, type1 = 0.005, root = 0)
```

Arguments

`alt`	a numeric; the value of hypothesized parameter (consistent with `side`) where the Type 2 error of the fixed design test needs to be evaluated at.
`side`	a character; direction of the alternative hypothesis H1. Has to be one of `"right"` or `"left"`. **Default:** `"right"`.
`n1`	a positive numeric (integer); sample size from Group-1 to be used.
`n2`	a positive numeric (integer); sample size from Group-2 to be used.

`type1`	a numeric in `(0,1)`; prespecified Type 1 error probability. **Default:** `0.005`.
`root`	a numeric; when this is 0, the Type 2 error is returned; in general, `(Type 2 error -root)` is returned **Default:** `0`.

Details

In case of two-sample T-tests in a fixed design, this function evaluates the Type 2 error at a specified value `alt`. We can also use this function to do the other way round by exploiting the argument `root`; that is, given a Type 2 error β, we can obtain the `alt` where Type 2 error equals β. To do that, we need to substitute `root=`β in the argument of this function, and then solve it for `alt`. The function `find.alt()` in this package excatly does this.

Value

If `root=k`, a numeric `(Type 2 error -k)` evaluated at the speified alternative value `alt` is returned.

Author(s)

Sandipan Pramanik, Valen E. Johnson and Anirban Bhattacharya

Examples

```
## Type 2 error at an alternative value
type2.error.twoT(alt= 1.2, n1= 60, n2= 60)

## (Type 2 error - 0.5) at the same alternative value
type2.error.twoT(alt= 1.2, n1= 60, n2= 60, root = 0.5)
```

---

type2.error.twoZ *Type 2 error function of two-sample Z-tests in a fixed design*

Description

For a two-sample Z-test in a fixed design, this function evaluates (Type 2 error - a constant) at a specified alternative value of the hypothesized parameter (the difference between the population means of Group-2 and Group-1). The hypothesized value under the simple null hypothesis is set at 0.

Usage

```
type2.error.twoZ(alt, side = "right", sigma0 = 1, n1, n2, type1 = 0.005, root = 0)
```

Arguments

`alt`	a numeric; the value of hypothesized parameter (consistent with `side`) where the Type 2 error of the fixed design test needs to be evaluated at.
`side`	a character; direction of the alternative hypothesis H1. Has to be one of `"right"` or `"left"`. **Default:** `"right"`.
`sigma0`	a positive numeric; the known common population standard deviation. **Default:** 1.
`n1`	a positive numeric (integer); sample size from Group-1 to be used.
`n2`	a positive numeric (integer); sample size from Group-2 to be used.
`type1`	a numeric in `(0,1)`; prespecified Type 1 error probability. **Default:** `0.005`.
`root`	a numeric; when this is 0, the Type 2 error is returned; in general, `(Type 2 error -root)` is returned **Default:** `0`.

Details

In case of two-sample Z-tests in a fixed design, this function evaluates the Type 2 error at a specified value `alt`. We can also use this function to do the other way round by exploiting the argument `root`; that is, given a Type 2 error β, we can obtain the `alt` where Type 2 error equals β. To do that, we need to substitute `root=`β in the argument of this function, and then solve it for `alt`. The function `find.alt()` in this package excatly does this.

Value

If `root=k`, a numeric `(Type 2 error -k)` evaluated at the speified alternative value `alt` is returned.

Author(s)

Sandipan Pramanik, Valen E. Johnson and Anirban Bhattacharya

Examples

```
## Type 2 error at an alternative value
type2.error.twoZ(alt= 1.2, n1= 60, n2= 60)

## (Type 2 error - 0.5) at the same alternative value
type2.error.twoZ(alt= 1.2, n1= 60, n2= 60, root = 0.5)
```

`ump.match.oneProp` *Finding the "evidence threshold (δ)" in one-sample proportion tests in a fixed design*

Description

In one-sample proportion tests in a fixed design, this function solves for δ by matching the rejection region from the UMPBT with that of the corresponding UMP (fixed design) test. Basically, this solves equation (20) in the supplemental information.

Usage

```
ump.match.oneProp(side = "right", type1 = 0.005, n, p0 = 0.5)
```

Arguments

`side`	a character; direction of the alternative hypothesis H1. Has to be one of `"right"` or `"left"`. **Default:** `"right"`.
`type1`	a numeric in `(0,1)`; prespecified Type 1 error probability. **Default:** `0.005`.
`n`	a positive numeric (integer); sample size to be used.
`p0`	a numeric in `(0,1)`; the hypothesized value of proportion under the simple null hypothesis. **Default** is `0.5`.

Author(s)

Sandipan Pramanik, Valen E. Johnson and Anirban Bhattacharya

References

MSPRT: Supplemental information

Johnson, Valen E., Uniformly most powerful Bayesian tests., Ann. of Stat., 41, (4), 2013, pp. 1716-1741

Examples

```
ump.match.oneProp(n= 60, p0= .2)
```

`umpbt.oneProp`	*The UMPBT alternative used in the MSPRT in one-sample proportion tests*

Description

This function finds the UMPBT alternative, a 2-points mixture distribution, used in a MSPRT design in one-sample proportion tests (Table 1 in the main article). This is obtained by matching the rejection region of the UMPBT to that of the randomized fixed design test. For more details please refer to the following references.

This is a slight modification of the UMPBT point alternative as originally defined in Johnson (2013). This point alternative can be calculated using `point.umpbt.oneProp()`.

Usage

```
umpbt.oneProp(side = "right", type1 = 0.005, n, null = 0.5)
```

Arguments

`side`	a character; direction of the alternative hypothesis H1. Has to be one of `"right"` or `"left"`. **Default:** `"right"`.
`type1`	a numeric in `(0,1)`; the prespecified Type 1 error probability. **Default:** `0.005`.
`n`	a positive integer; number of samples to be used.
`null`	a numeric in `(0,1)`; the hypothesized value of proportion under the simple null hypothesis. **Default** is `0.5`.

Details

The UMPBT alternative used in the MSPRT in one-sample proportion tests is a mixture distribution with two points. This function returns those two points and the mixing probability.

Value

Returns a list with the following two components:

`u`	a numeric vector of length 2; these are the two points of the mixture distribution.
`psi`	a numeric in `(0,1)`; mixing probability corresponding to the first component of `u`.

Author(s)

Sandipan Pramanik, Valen E. Johnson and Anirban Bhattacharya

References

MSPRT: main article and supplemental information

Johnson, Valen E., Uniformly most powerful Bayesian tests., Ann. of Stat., 41, (4), 2013, pp. 1716-1741

Johnson, Valen E., Revised standards for statistical evidence., Proceedings of the National Academy of Sciences, 16, 1945. (Specifically, it's supplemental file)

Examples

```
# the point UMPBT alternative according to Johnson (2013)
point.umpbt.oneProp(n= 60, null= .2)
point.umpbt.oneProp(side= "left", n= 60, null= .2)

# the UMPBT alternative used by the MSPRT in these cases
umpbt.oneProp(n= 60, null= .2)
umpbt.oneProp(side= "left", n= 60, null= .2)
```

`umpbt.oneT` *The UMPBT alternative in one-sample T-tests*

Description

This function finds the approximate data dependent UMPBT alternative (Table 1 in the main article) in one-sample T-tests. For more details please refer to the following references.

Usage

```
umpbt.oneT(side = "right", type1 = 0.005, n, null = 0, obs, s)
```

Arguments

`side`	a character; direction of the alternative hypothesis H1. Has to be one of `"right"` or `"left"`. **Default:** `"right"`.
`type1`	a numeric in `(0,1)`; prespecified Type 1 error probability. **Default:** `0.005`.
`n`	a positive numeric (integer); sample size to be used.
`null`	a numeric; the hypothesized value of population mean under the simple null hypothesis. **Default:** `0`.
`obs`	a numeric vector; the vector of observations based on which the alternative needs to be calculated. This is ignored if `s` is provided.
`s`	a positive numeric; the sample standard deviation (sd) (of divisor `(n-1)`) of the `obs`. Can be missing if `obs` is provided.

Details

We need either `obs` or `s`.

When we are implementing the MSPRT in one-sample T-test and we need the alternative at Step-5, this function requires all the data observed until Step-5 in the order they were observed.

Value

Returns a numeric which is the UMPBT alternative in the one-sample T-test.

Author(s)

Sandipan Pramanik, Valen E. Johnson and Anirban Bhattacharya

References

MSPRT: main article and supplemental information

Johnson, Valen E., Uniformly most powerful Bayesian tests., Ann. of Stat., 41, (4), 2013, pp. 1716-1741

Johnson, Valen E., Revised standards for statistical evidence., Proceedings of the National Academy of Sciences, 16, 1945. (Specially it's supplemental file)

Examples

```
# a simulated ordered data at step-30
x.seq = rnorm(30,2,1.5)

# UMPBT alternative at step-30

## providing the data x.seq
umpbt.oneT(n= 60, obs= x.seq)

## providing the sd of x.seq
umpbt.oneT(n= 60, s= sd(x.seq))
```

`umpbt.oneZ` *The UMPBT alternative in one-sample Z-tests*

Description

This function finds the UMPBT alternative (Table 1 in the main article) in one-sample Z-tests. This is obtained by matching the rejection region of the UMPBT to that of the fixed design test. For more details please refer to the following references.

Usage

```
umpbt.oneZ(side = "right", type1 = 0.005, n, null = 0, sigma0 = 1)
```

Arguments

`side`	a character; direction of the alternative hypothesis H1. Has to be one of `"right"` or `"left"`. **Default:** `"right"`.
`type1`	a numeric in `(0,1)`; prespecified Type 1 error probability. **Default:** `0.005`.
`n`	a positive numeric (integer); sample size to be used.
`null`	a numeric; the hypothesized value of population mean under the simple null hypothesis. **Default:** `0`.
`sigma0`	a positive numeric; the known population standard deviation. **Default:** 1.

Value

Returns a numeric which is the UMPBT point alternative in the one-sample Z-test.

Author(s)

Sandipan Pramanik, Valen E. Johnson and Anirban Bhattacharya

References

MSPRT: main article and supplemental information

Johnson, Valen E., Uniformly most powerful Bayesian tests., Ann. of Stat., 41, (4), 2013, pp. 1716-1741

Johnson, Valen E., Revised standards for statistical evidence., Proceedings of the National Academy of Sciences, 16, 1945. (Specially it's supplemental file)

Examples

```
umpbt.oneZ(n= 60)
```

`umpbt.twoT` *The UMPBT alternative in two-sample T-tests*

Description

This function finds the approximate data dependent UMPBT alternative in two-sample T-tests. The value of hypothesized parameter, the difference between the population means of Group-2 and Group-1, under the simple null hypothesis is set at 0.

Usage

```
umpbt.twoT(side = "right", type1 = 0.005, n1, n2, obs1, obs2, s)
```

Arguments

`side`	a character; direction of the alternative hypothesis H1. Has to be one of `"right"` or `"left"`. **Default:** `"right"`.
`type1`	a numeric in `(0,1)`; prespecified Type 1 error probability. **Default:** `0.005`.
`n1`	a positive numeric (integer); sample size from Group-1 to be used.
`n2`	a positive numeric (integer); sample size from Group-2 to be used.
`obs1`	a numeric vector; the vector of observations from Group-1 based on which the alternative is calculated.
`obs2`	a numeric vector; the vector of observations from Group-2 based on which the alternative is calculated.
`s`	a positive numeric; the pooled sample standard deviation (sd) based on `obs1` and `obs2`. Can be missing if `obs1` and `obs2` are provided.

Details

We need either `obs1` & `obs2`, or `s`.

When we are implementing the MSPRT in a two-sample T-test and we need the alternative at Step-5, this function requires all the data from both groups observed until that step in the order they were observed.

Value

Returns a numeric which is the UMPBT alternative in the two-sample T-test.

Author(s)

Sandipan Pramanik, Valen E. Johnson and Anirban Bhattacharya

References

MSPRT: main article and supplemental information

Johnson, Valen E., Uniformly most powerful Bayesian tests., Ann. of Stat., 41, (4), 2013, pp. 1716-1741

Johnson, Valen E., Revised standards for statistical evidence., Proceedings of the National Academy of Sciences, 16, 1945. (Specially it's supplemental file)

Examples

```
# suppose we get data from both groups after each observation
# a simulated ordered data at Step-30
x1.seq = rnorm(30,2,1.5)
x2.seq = rnorm(30,2,1.5)
```

```
## UMPBT alternative at step-30
# providing the data x1.seq & x2.seq
umpbt.twoT(n1= 60, n2= 60, obs1= x1.seq, obs2= x2.seq)
```

`umpbt.twoZ` *The UMPBT alternative in two-sample Z-tests*

Description

This function finds the UMPBT alternative in two-sample Z-tests. The value of hypothesized parameter, the difference between the population means of Group-2 and Group-1, under the simple null hypothesis is set at 0.

Usage

```
umpbt.twoZ(side = "right", type1 = 0.005, n1, n2, sigma0 = 1)
```

Arguments

`side`	a character; direction of the alternative hypothesis H1. Has to be one of `"right"` or `"left"`. **Default:** `"right"`.
`type1`	a numeric in `(0,1)`; prespecified Type 1 error probability. **Default:** `0.005`.
`n1`	a positive numeric (integer); sample size from Group-1 to be used.
`n2`	a positive numeric (integer); sample size from Group-2 to be used.
`sigma0`	a positive numeric; the known common population standard deviation. **Default:** 1.

Value

Returns a numeric which is the UMPBT alternative in the two-sample Z-test.

Author(s)

Sandipan Pramanik, Valen E. Johnson and Anirban Bhattacharya

References

MSPRT: main article and supplemental information

Johnson, Valen E., Uniformly most powerful Bayesian tests., Ann. of Stat., 41, (4), 2013, pp. 1716-1741

Johnson, Valen E., Revised standards for statistical evidence., Proceedings of the National Academy of Sciences, 16, 1945. (Specially it's supplemental file)

Examples

```
umpbt.twoZ(n1= 60, n2= 60)
```

Index