Package 'MSPRT'

September 17, 2018

Type Package

Title Modified Sequential Probability Ratio Test (MSPRT)

Version 1.0

Date 2018-09-11

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, the maximum sample size of an experiment is fixed prior to the start of an experiment, 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 the targeted Type 2 error (equal 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 levels of significance and power. This package facilitates the carrying out of one sample Z tests, T Tests and tests of binomial success probabilities. A user guidance for this software package is provided here and also in the supplemental information.

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

License GPL (>= 2)

NeedsCompilation no

Repository CRAN

Date/Publication 2018-09-17 14:50:03 UTC

R topics documented:

`MSPRT-package` *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, the maximum sample size of an experiment is fixed prior to the start of an experiment, 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 the targeted Type 2 error (equal 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 levels of significance and power. This package facilitates the carrying out of one sample Z tests, T Tests and tests of binomial success probabilities. A user guidance for this software package is provided here and also in the supplemental information.

Details

Package:	MSPRT
Type:	Package
Version:	1.0
Date:	09-11-2018
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. If it stays inconclusive after maximum number of comparisons, then it compares with a threshold which is just like the termination threshold in a MSPRT.

Usage

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

Arguments

`test.type`	a character; denotes the type of test; "`prop.test`" in case of a test for binomial proportion "`z.test`" in case of a Z-test "`t.test`" in case of a 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 obtained likelihood ratios (L_n); it's length should not exceed the length of `batch.seq`
`upper`	a numeric vector; the sequence of upper threshold; it's length should equal to the length of `batch.seq`
`lower`	a numeric vector; the sequence of lower threshold; it's length should equals to the length of `batch.seq`
`batch.seq`	a numeric vector; an increasing sequence of sample size values where data are supposed to be observed sequentially
`threshold`	a positive numeric; similar to the termination threshold in a MSPRT

Details

Suppose a researcher can affrod at most 100 samples, and can only observe every 10th sample. Then `batch.seq` will be `seq( from=10, to=100, by=10)`. So there are at most 10 batches/stages where we can observe the data and make compare.

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

Not to mention, any element of the `lower` cannot be greater than the corresponding element of the `upper`.

Value

Returns a list containing the following elements:

`decision`	a character; the final decision that is made; either `"accept"`, `"reject"` or `"continue"`
`n`	a numeric; how many samples were needed for coming to 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 both of them componentwise.

Usage

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

Arguments

`l1`	a list
`l2`	a list

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 modified Sequential Probability Ratio Test (MSPRT)*

Description

Given desired values of Type 1 & Type 2 error probabilities and the maximum number of samples (N), this function designs the MSPRT (by finding the 'Termination Threshold' γ). γ is chosen as 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 3 one sample tests: (1) Test for a binomial proportion, (2) Z-test, and (3) T-test.

By default, this provides the operating characteristics (OC) for the obtained MSPRT at the null hypothesized value. It can also find the same at a user desired point alternative which we can specify through `alt.comp`. In general, `OC.MSPRT()` can be used to find the OC of a MSPRT at any desired parameter value.

Usage

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

Arguments

`test.type`	a character; determines the type of test; "`prop.test`" in case of a test for binomial proportion; "`z.test`" in case of a Z-test; "`t.test`" in case of a T-test.
`side`	a character; direction of the alternative hypothesis H1; has to be one of `"right"` or `"left"`. **Default** is `"right"`.
`batch.seq`	a numeric vector; An increasing sequence of values until `N.max`. Denotes the sequence of sample sizes where a user will observe data sequentially. Last element should equal to `N.max`. **Default:** In case of the randomized test for binomial proportion and the Z-test, this is `1:N.max`. The same for T-test is `2:N.max`.
`type1`	a numeric in `(0,1)`; The probability at which we want to control the Type 1 error of the MSPRT. **Default** is `0.005`.

`type2` a numeric in `(0,1)`;
The probability at which we want to control the Type 2 error of the MSPRT.
Default is `0.2`.

`null` a numeric;
The hypothesized value of the parameter under the null hypothesis.
The hypothesized parameters are proportion in case of testing a binomial proportion, and population mean in case of Z and T tests.
Default: 0.5 in case of the test for binomial proportion, and 0 for Z and T-tests.

`sigma0` a positive numeric;
the known standard deviation (sd) in a Z-test. Only need to be specified when carrying out a Z-test.
Default is 1.

`N.max` a positive numeric (integer); maximum number of samples that we can afford.

`alt.comp` missing, `TRUE` or a numeric;
If missing, the OC of the MSPRT are obtained only at the `null`;
If `TRUE`, the OC of the MSPRT are computed at the 'fixed design alternative';
If a numeric, it can be any value under the alternative (consistent with `"side"`). Then OC of the MSPRT are obtained at this point.

`repl` a positve numeric (integer); total no. of replications to be used in Monte Carlo method to calculate the OC for an MSPRT;
Default: `2e+6` for a proportion test; `1e+6` for Z or T tests;
should be at least `1e+5`.

`verbose` logical; `TRUE` or `FALSE`;
If `TRUE`, returns messages of the current proceedings; otherwise it doesn't;
Default is `TRUE`.

`core.no` a numeric;
Number of cores this function can use for carrying out this computation using the parallel computing.
Default is 1 if there is at most 2 cores, otherwise `(number of cores -1)`

Details

At the least, we need to provide `test.type` and `N.max`. For a brief user guide, please refer to the main article and the supplemental information.

Value

If `alt.comp` is missing (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;

For the proportion test, this is usually of length 2. They specify the two points of the UMPBT alternative.

For the Z-test, this is the UMPBT point alternative.

For T-test, this is not returned.

`psi.umpbt` a numeric in `(0,1)`;

Returned only in case of the proportion test. 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.

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 'fixed design alternative'.

`avg.n1` a positive numeric; the number of samples required on an average by the MSPRT for coming to a decision when the 'fixed design alternative' is actually 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 as was for `alt.comp = TRUE` with `type2.est`, `avg.n1`, `alt` and `alt.type2` now computed at this specified value.

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

```
## test for a binomial proportion

# can sequentially observe every sample where max available
# sample size is 30
#design.MSPRT(test.type="prop.test", side = "right", null = 0.2, N.max = 30)


# can sequentially observe every fifth sample where max available
# sample size is 30
#design.MSPRT(test.type="prop.test", side = "right", null = 0.2, N.max = 30,
#             batch.seq = seq(5,30, by = 5))



## Z-test

# can sequentially observe every sample where max available
# sample size is 30
#design.MSPRT(test.type="z.test", side = "right", null = 3, sigma0 = 1.5,
#             N.max = 30)


# can sequentially observe every fifth sample where max available
# sample size is 30
#design.MSPRT(test.type="z.test", side = "right", null = 3, sigma0 = 1.5,
#             N.max = 30, batch.seq = seq(5,30, by = 5))



## T-test

# can sequentially observe every sample where max available
# sample size is 30
#design.MSPRT(test.type="t.test", side = "right", null = 3, N.max = 30)


# can sequentially observe every fifth sample where max available
# sample size is 30
#design.MSPRT(test.type="t.test", side = "right", null = 3, N.max = 30,
#             batch.seq = seq(5,30, by = 5))
```

`effective.N` *Effective maximum sample size for a MSPRT in a proportion test*

Description

Because of the discreteness issue in a proportion test, it may not be effective if we use just any value as a maximum sample size to design a MSPRT. Given a desired value for the maximum sample size,

this function finds the maximum sample size which should be used for the design.

Usage

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

Arguments

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

Details

First we shortlist only those values within N.max at which the UMPBT point alternative, as is originally defined in Johnson (2013), strictly decreases. The final value is chosen to be the maximum among those shorlisted value.

Value

Returns a numeric. This is the 'effective' maximum sample size which should be used to design a MSPRT.

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.max = 30, null.val = .2)
```

`error.summary` *Error probability of a MSPRT*

Description

After any of the functions `overshoot.ber()`, `overshoot.norm()` or `overshoot.t()` is executed, this function post processes it's outputs and then calculates the Type 1 (or Type 2) error probability of a MSPRT given a value of the termination threshold.

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; a possible value of the termination threshold
`root`	a numeric; based on this (error probability - `root`) is returned
`count`	a numeric (integer); same as the "`count`" output from `overshoot.ber()`, `overshoot.norm()` or `overshoot.t()` with the same `error.type`
`inconclusive.vec`	a numeric vector; same as the "`inconclusive.vec`" output from `overshoot.ber()`, `overshoot.norm()` or `overshoot.t()` with the same `error.type`
`R`	a numeric (integer); number of replications in Monte Carlo method that is used in `overshoot.ber()`, `overshoot.norm()` or `overshoot.t()` with the same `error.type`
`type1`	a numeric in `(0,1)`; the prespecified Type 1 error probability used in `overshoot.ber()`, `overshoot.norm()` or `overshoot.t()`
`type2`	a numeric in `(0,1)`; the prespecified Type 2 error probability used in `overshoot.ber()`, `overshoot.norm()` or `overshoot.t()`

Details

Suppose we have the Monte Carlo method outputs from `overshoot.ber()`, `overshoot.norm()` or `overshoot.t()` corresponding to some `error.type`. Given this, the function calculates the difference (error probability - `root`) of a MSPRT where `delta` is used as a value of the termination threshold.

Value

a numeric; the difference (estimated error probability - `root`)

Author(s)

Sandipan Pramanik, Valen E. Johnson and Anirban Bhattacharya

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

Description

This function finds the 'fixed design alternative' in case of a test for binomial proportion, Z and T-test.

Usage

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

Arguments

`test.type`	a character; "`prop.test`" in case of a test for binomial proportion "`z.test`" in case of a Z-test "`t.test`" in case of a T-test
`side`	a character; direction of the alternative hypothesis H1; has to be one of `"right"` or `"left"`
`null`	a numeric; denotes the hypothesized value under the null hypothesis; The hypothesized parameters are proportion in case of testing a binomial proportion, and population mean in case of Z and T tests. has to be in `(0,1)` in case of a proportion test, but can be any real value in case of a Z or T Test
`size`	a positive numeric (integer); number of samples the fixed design test is based on
`type1`	a numeric in `(0,1)`; desired Type 1 error probability of the fixed design test
`type2`	a numeric in `(0,1)`; desired Type 2 error probability corresponding to which we want to obtain the 'fixed design alternative'
`sigma0`	a positive numeric; denotes the known standard deviation in a Z-test; need to be specified only if `test.type = "z.test"`

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 proportion test which
## provides 80% power against a right and left sided
## alternative, respectively

# default null = 0.5
find.alt(test.type="prop.test", side= "right", size= 60,
         type1= 0.005, type2= 0.2)

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

# null = 0.2
find.alt(test.type="prop.test", side= "right", null= 0.2,
         size= 60, type1= 0.005, type2= 0.2)

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


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

# default null = 0
find.alt(test.type="z.test", side= "right",
         size= 60, type1= 0.005, type2= 0.2, sigma0= 1)

find.alt(test.type="z.test", side= "left",
         size= 60, type1= 0.005, type2= 0.2, sigma0= 1)

# null = 3
find.alt(test.type="z.test", side= "right", null= 3,
         size= 60, type1= 0.005, type2= 0.2, sigma0= 1)

find.alt(test.type="z.test", side= "left", null= 3,
         size= 60, type1= 0.005, type2= 0.2, sigma0= 1)


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

# default null = 0
find.alt(test.type="t.test", side= "right", size= 60,
         type1= 0.005, type2= 0.2)

find.alt(test.type="t.test", side= "left", size= 60,
```

```
         type1= 0.005, type2= 0.2)

# null = 3
find.alt(test.type="t.test", side= "right", null= 3,
         size= 60, type1= 0.005, type2= 0.2)

find.alt(test.type="t.test", side= "left", null= 3,
         size= 60, type1= 0.005, type2= 0.2)
```

`find.samplesize` *Sample size required to achieve a higher significance in a fixed design*

Description

This function finds the sample size that is required when we decrease the level of significance of a test while still maintaining a desired power at a point alternative. The sample size can be obtained in case of the test for proportion, Z-test and T-test.

Usage

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

Arguments

`test.type`	a character; determines the type of test; "`prop.test`" in case of a test for binomial proportion; "`z.test`" in case of a Z-test; "`t.test`" in case of a T-test.
`N`	a numeric; sample size used at the lower level of significance in a fixed design
`lower.signif`	a numeric in `(0,1)`; denotes the lower significance level **Default** is `0.05`.
`higher.signif`	a numeric in `(0,1)`; denotes the higher significance level **Default** is `0.005`.
`null.val`	a numeric; The hypothesized value of the parameter under the null hypothesis. The hypothesized parameters are proportion in case of testing a binomial proportion, and population mean in case of Z and T tests. **Default:** 0.5 in case of the test for binomial proportion, and 0 for Z and T-tests.
`side`	a character; direction of the alternative hypothesis H1; has to be one of `"right"` or `"left"`. **Default** is `"right"`.

pow	a numeric in (0,1); desired level of power at the point alternative **Default** is 0.8, means 80 percent power.
alt	missing or a numeric; a value of the point alternative where we want to maintain the power; **Default:** the fixed design alternative at the lower.signif using N samples.
sigma0	a positive numeric; Specifies the known standard deviation (sd) in a Z-test; Only need to be specified in case of a Z-test; **Default** is 1.
n.seq	missing or a numeric vector; final value of the sample size is searched over this vector. **Default** is N:(4*N).
plot.it	a logical; if TRUE, returns a plot. Otherwise not. **Default** is TRUE.

Value

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

Author(s)

Sandipan Pramanik, Valen E. Johnson and Anirban Bhattacharya

References

Main article and supplemental file of MSPRT

Examples

```
find.samplesize(test.type = "prop.test", N = 30, null.val = .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 = "prop.test", N = 30, null.val = .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.thresold.ber *Optimizing the UMPBT objective function in case of a proportion test*

Description

Given δ, this function finds the difference (optimum value of the objective function - a constant) in case of a test for binomial proportion. Notation is similar to the supplemental information.

Usage

```
find.thresold.ber(delta, side = "right", n.obs, p0 = 0.5, opt.interval, root)
```

Arguments

delta	a positive numeric corresponding to which 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** is "right"
n.obs	a positive integer; number of samples used in the fixed design test
p0	a numeric in (0,1); hypothesized value of the 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** is c(p0,1) if side = "right", and c(0,p0) if side = "left".
root	a numeric; the constant in the above description

Details

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

Value

If root=k, given the delta this returns a numeric denoting the difference (optimum value of the objective function -

Author(s)

Sandipan Pramanik, Valen E. Johnson and Anirban Bhattacharya

Examples

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

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

`find.umpbt.ber` *The UMPBT alternative used in the MSPRT in case of a proportion test*

Description

This function finds the UMPBT alternative, the 2-points mixture distribution, used by the MSPRT design in case of a proportion test. 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 supplemental information.

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

Usage

```
find.umpbt.ber(side = "right", type1 = 0.005, n.obs, null = 0.5)
```

Arguments

`side`	a character; direction of the alternative hypothesis H1; has to be one of `"right"` or `"left"`; **Default** is `"right"`
`type1`	a numeric in `(0,1)`; the prespecified Type 1 error **Default** is `0.005`
`n.obs`	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 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.(Specially it's supplemental file)

Examples

```
find.umpbt.ber(side= "right", n.obs= 60, null= .2)
find.umpbt.ber(side= "left", n.obs= 60, null= .2)
```

---

find.umpbt.norm *The UMPBT alternative in a Z-test*

Description

This function finds the UMPBT point alternative (Table 1 of the main article) in case of a Z-test. 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 supplemental information.

Usage

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

Arguments

side	a character; direction of the alternative hypothesis H1; has to be one of "right" or "left"; **Default** is "right"
type1	a numeric in (0,1); the prespecified Type 1 error; **Default** is 0.005
n.obs	a positive integer; number of samples to be used
null	a numeric; hypothesized value of population mean under the simple null hypothesis **Default** is 0
sigma0	a positive numeric; the known standard deviation (sd) in the Z-test **Default** is 1

Value

Returns a real numeric which is the UMPBT point alternative in a 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

```
find.umpbt.norm(n.obs= 60)
```

`find.umpbt.t` *The UMPBT alternative in a T-test*

Description

This function finds the approximate data dependent UMPBT point alternative (Table 1 of the main article) in case of a T-test. For more details please refer to the supplemental information.

Usage

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

Arguments

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

obs	a numeric vector; a vector of the observations based on which the alternative is calculated; If the alternative is required at step 5 when we are implementing a MSPRT, we need to provide all the data observed until that step in the order they were observed; can be missing 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 at least one of obs or s.

Value

Returns a real numeric which is the UMPBT alternative in the 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
find.umpbt.t(n.obs= 60, obs= x.seq)

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

`implement.MSPRT` *Implementing a modified Sequential Probability Ratio Test (MSPRT)*

Description

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

Usage

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

Arguments

`obs`	a numeric vector; denotes the sequentially observed data based on which we want to carry out the null hypothesis significance testing (NHST) This is a vector of all the observed data (until the present) in the order they are observed.
`test.type`	a character; determines the type of test; "`prop.test`" in case of a test for binomial proportion; "`z.test`" in case of a Z-test; "`t.test`" in case of a T-test.
`side`	a character; direction of the alternative hypothesis H1; has to be one of `"right"` or `"left"`. **Default** is `"right"`.
`batch.seq`	a numeric vector; An increasing sequence of values until `N.max`. Denotes the sequence of sample sizes where a user will observe data sequentially. Last element should equal to `N.max`. **Default:** In case of the randomized test for binomial proportion and the Z-test, this is `1:N.max`. The same for T-test is `2:N.max`.
`type1`	a numeric in `(0,1)`; The probability at which we want to control the Type 1 error of the MSPRT. **Default** is `0.005`.
`type2`	a numeric in `(0,1)`; The probability at which we want to control the Type 2 error of the MSPRT. **Default** is `0.2`.

null	a numeric; The hypothesized value of the parameter under the null hypothesis. The hypothesized parameters are proportion in case of testing a binomial proportion, and population mean in case of Z and T tests. **Default:** 0.5 in case of the test for binomial proportion, and 0 for Z and T-tests.
sigma0	a positive numeric; the known standard deviation (sd) in a Z-test. Only need to be specified when carrying out a Z-test. **Default** is 1.
term.thresh	a positive numeric; the termination threshold; this is obtained from design.MSPRT().
N.max	a positive numeric (integer); maximum number of samples that we can afford.
plot.it	logical vector; if TRUE (**Default**) it returns a comparison plot. Otherwise it doesn't.
verbose	logical; TRUE or FALSE; If TRUE, returns messages of the current proceedings; otherwise it doesn't; **Default** is TRUE.

Details

Suppose we want to carry out one of the above one sample tests. We need to follow two steps to carry out a MSPRT for this.

Step 1: Designing the MSPRT: Using design.MSPRT() we determine the 'Termination threshold'.

Step 2: Implementing the MSPRT: Using implement.MSPRT() we implement the MSPRT algorithm for a sequentially observed data. The termination threshold obtained from Step 1 is used here.

Value

Returns a list with the following components:

decision	a character; the decision reached based on the provided data obs; 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 N.max.
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 a proportion test; returned only in case of a proportion test.

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()

##test for a binomial proportion

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 = "prop.test", 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 = "z.test", 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 = "t.test", null = 3,
                term.thresh = 32.702, N.max = 30)
```

LR.ber *Likelihood ratio in a test for binomial proportion*

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 case of a test for binomial proportion.

Usage

```
LR.ber(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); value of the 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)`; hypothesized value of the binomial proportion under the simple null **Default** is 0.5.
`alt`	a numeric in `(0,1)`; hypothesized value of the binomial proportion under the simple alternative

Value

Returns a numeric denoting the LR in favor of the `alt` in the test for binomial proportion based on `m` observations.

Author(s)

Sandipan Pramanik, Valen E. Johnson and Anirban Bhattacharya

References

MSPRT: supplemental information

Examples

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

LR.norm *Likelihood ratio in a Z-test*

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 case of a Z-test.

Usage

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

Arguments

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

Value

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

Author(s)

Sandipan Pramanik, Valen E. Johnson and Anirban Bhattacharya

References

MSPRT: supplemental information

Examples

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

LR.t *Bayes factor in a T-test*

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 case of a T-test.

Usage

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

Arguments

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

Value

Returns a numeric denoting the BF in favor of the specified alt in case of T-test based on m observations.

Author(s)

Sandipan Pramanik, Valen E. Johnson and Anirban Bhattacharya

References

MSPRT: supplemental information

Examples

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

`obj.func.ber` *Objective function for determining the UMPBT point alternative in a proportion test*

Description

The $h(p, \delta)$ function as in the supplemental file. Given a δ, we get the UMPBT(δ) alternative by optimizing this function.

Usage

```
obj.func.ber(p, delta, n.obs, p0)
```

Arguments

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

```
obj.func.ber(p= .5, delta= 25, n.obs= 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 this 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 the cases this also computes the number of samples those are required on an average for terminating the sampling and reaching a decision.

Usage

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

Arguments

`test.type`	a character; determines the type of test; "`prop.test`" in case of a test for binomial proportion; "`z.test`" in case of a Z-test; "`t.test`" in case of a T-test.
`side`	a character; direction of the alternative hypothesis H1; has to be one of `"right"` or `"left"`. **Default** is `"right"`.
`batch.seq`	a numeric vector; An increasing sequence of values until `N.max`. Denotes the sequence of sample sizes where a user will observe data sequentially. Last element should equal to `N.max`. **Default:** In case of the randomized test for binomial proportion and the Z-test, this is `1:N.max`. The same for T-test is `2:N.max`.
`null`	a numeric; The hypothesized value of the parameter under the null hypothesis. The hypothesized parameters are proportion in case of testing a binomial proportion, and population mean in case of Z and T tests. **Default:** 0.5 in case of the test for binomial proportion, and 0 for Z and T-tests.
`term.thresh`	a positive numeric; denotes the termination threshold of a MSPRT; this is determined at the designing step of a MSPRT by using `design.MSPRT()`.
`theta`	a numeric; denotes the hypothesized parameter value where we want to evaluate the OC for the MSPRT.
`sigma0`	a positive numeric; the known standard deviation (sd) in a Z-test. Only need to be specified when carrying out a Z-test. **Default** is 1.

`type1` a numeric in `(0,1)`;
The probability at which we want to control the Type 1 error of the MSPRT.
Default is `0.005`.

`type2` a numeric in `(0,1)`;
The probability at which we want to control the Type 2 error of the MSPRT.
Default is `0.2`.

`N.max` a positive numeric (integer); maximum number of samples that we can afford.

`verbose` logical; `TRUE` or `FALSE`;
If `TRUE`, returns messages of the current proceedings; otherwise it doesn't;
Default is `TRUE`.

`repl` a positve numeric (integer); total no. of replications to be used in Monte Carlo method to calculate the OC for an MSPRT;
Default: `2e+6` for a proportion test; `1e+6` for Z or T tests;
should be at least `1e+5`.

`core.no` a numeric;
Number of cores this function can use for carrying out this computation using the parallel computing.
Default is `1` if there is 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 in the region of values under the alternative hypothesis, then Type 2 error and `avg.n1` are calculated; otherwise Type 1 error and `avg.n0` are calculated.

Value

Returns a list with the following components:

`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` .

Author(s)

Sandipan Pramanik, Valen E. Johnson and Anirban Bhattacharya

Examples

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

## test for a single proportion

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



## z-test
## finding OC at theta  = 4

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


## t-test

#OC.MSPRT(test.type = "t.test", null = 3, term.thresh = 32.702,
#         theta = 4, N.max = 30)
```

`overshoot.ber` *Error summary of Wald's SPRT truncated at the maximum available sample size in a proportion test*

Description

MSPRT is designed on the assumption that a researcher can afford at most, say, `N` samples. This function calculates Type 1 or Type 2 error summaries committed by the Wald's SPRT when it is simply truncated at `N` in a proportion test.

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

Usage

```
overshoot.ber(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 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 binomial proportion from which the data needs to be generated from

`alt.LR` a numeric vector of lenth 2;

this is a vector of the 2-points of the UMPBT alternative; the sequence of weighted likelihood ratios (L_n) is computed in favour of this alternative;

this is same with the output `u` from `find.umpbt.ber`.

`alt.psi` a numeric in `(0,1)`;

denotes the mixing probability corresponding to the first component of `alt.LR` in the UMPBT alternative;

this is same with the output `psi` from `find.umpbt.ber()`.

`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 case of 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` out of `R` replications those are committed

`inconclusive.vec`

a numeric vector; a vector containing values of L_N which remain inconcluded after the truncation

`n.vec` a numeric vector; a vector of required number of samples; this is of length `R`.

If `return.n = FALSE`, the same list with all the components 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.ber( 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.ber( 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.norm — *Error summary of Wald's SPRT truncated at the maximum available sample size in a Z-test*

Description

MSPRT is designed on the assumption that a researcher can afford at most, say, N samples. This function calculates Type 1 or Type 2 error summaries committed by the Wald's SPRT when it is simply truncated at N in a Z-test.

It is worth a mention that a case may remain inconcluded due to the truncation. The required sample size for reaching a decision in those cases is N.

Usage

```
overshoot.norm(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 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 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; observations are generated from a normal distribution with these specifications
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 find.umpbt.norm().
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 case of 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 out of R replications those are committed
inconclusive.vec	a numeric vector; a vector containing values of L_N which remain inconcluded after the truncation
n.vec	a numeric vector; a vector of required number of samples; this is of length R.

If return.n = FALSE, the same list with all the components 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.norm( 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.norm( 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.t` *Error summary of Wald's SPRT truncated at the maximum available sample size in a T-test*

Description

MSPRT is designed on the assumption that a researcher can at most use, say `N`, samples. This function calculates Type 1 or Type 2 error summaries committed by the Wald's SPRT when it is simply truncated at `N` in case of a T-test.

It is worth a mention that a case may remain inconcluded due to the truncation. In those cases the required sample size is `N`.

Usage

```
overshoot.t(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 until `N`. Denotes the sequence of sample sizes where a user will observe data sequentially. Last element should equal to `N`.
`type1`	a numeric in `(0,1)`; The probability at which the user wants 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
low	a numeric; value of a constant acceptance threshold
N	a positive numeric (integer); number of samples where truncation of Wald's SPRT is required
R	a positive numeric (integer); number of replications desired in Monte Carlo method; at least 1e+5 is required
core.no	a numeric; number of cores this function can use for carrying out required Monte Carlo computation using parallel computing
return.n	logical; TRUE or FALSE if TRUE, corresponding to the cases where Wald's SPRT either accepts or rejects null before truncation, this function will return a vector of number of required samples; otherwise not

Value

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

count	a numeric; number of errors of error.type out of R replications those are committed
inconclusive.vec	a numeric vector; a vector containing values of L_N which remain inconcluded after the truncation
n.vec	a numeric vector; a vector of required number of samples

If return.n = FALSE, the same list with all the components 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.t( side="right", error.type= "type1", batch.seq= 1:N.max, null= 0,
#             gen.par= 0, up= 160, low= 0.2, N= N.max,
#             R= 1e+6, core.no= 2, return.n = T)

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

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

Description

In case of a proportion test this function generates a data, computes the weighted likelihood ratios and compares with the acceptance and rejection thresholds. Based on this, `overshoot.ber()` carries out a Monte Carlo method by repeating this function for, say, `R` number of times.

Usage

```
ovr.repl.ber(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 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 binomial proportion from which the data needs to be generated from
`alt.LR`	a numeric vector of lenth 2; this is the 2-points UMPBT alternative in favour of which the sequence of LR or bayes factors (L_n) need to be computed; similar to the output `u` from `find.umpbt.ber()`
`alt.psi`	a numeric in `(0,1)`; denotes the mixing probability corresponding to the first component of `alt.LR` in the mixture distribution like the UMPBT alternative; similar to the output `psi` from `find.umpbt.ber`
`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 case of 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; if Wald's SPRT either accepts or rejects null hypothesis at or before `N`, number of samples required for making that decision is returned; otherwise `N` 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
ovr.repl.ber( 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.ber( 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.norm *A particular replication step in* `overshoot.norm()`

Description

In case of a Z-test, this function generates a data, computes likelihood ratios and compares with acceptance and rejection thresholds. Based on this, `overshoot.norm()` carries out a Monte Carlo method by repeating this function for, say, `R` number of times.

Usage

```
ovr.repl.norm(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 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; observations are generated from a normal distribution with these specifications
alt.LR	a numeric; this is the alternative in favour of which the sequence of likelihood ratios(L_n) need to be computed; The UMPBT alternative is used in a MSPRT.
up	a numeric; value of a constant rejection threshold
low	a numeric; value of a constant acceptance threshold
N	a positive numeric (integer); number of samples where truncation of Wald's SPRT is required
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; if Wald's SPRT either accepts or rejects null hypothesis at or before N, number of samples required for making that decision is returned; otherwise N 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
ovr.repl.norm( 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.norm( 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.t` *A particular replication step in* `overshoot.t()`

Description

In case of a T-test, this function generates a data, computes likelihood ratios and compares with acceptance and rejection thresholds. Based on this, `overshoot.t()` carries out a Monte Carlo method by repeating this function for, say, `R` number of times.

Usage

```
ovr.repl.t(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 until `N`. Denotes the sequence of sample sizes where a user will observe data sequentially. Last element should equal to `N`.
`type1`	a numeric in `(0,1)`; The probability at which the user wants 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
`low`	a numeric; value of a constant acceptance threshold
`N`	a positive numeric (integer); number of samples where truncation of Wald's SPRT is required
`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; if Wald's SPRT either accepts or rejects null hypothesis at or before `N`, number of samples required for making that decision is returned; otherwise `N` 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
ovr.repl.t( 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.t( 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)
```

---

point.umpbt.ber *The UMPBT point alternative in case of a proportion test*

Description

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

`find.umpbt.ber()` is a slight modification of this.

Usage

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

Arguments

side	a character; direction of the alternative hypothesis H1; has to be one of "right" or "left"; **Default** is "right"
type1	a numeric in (0,1); the prespecified Type 1 error **Default** is 0.005
n.obs	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.ber(n.obs= 60)
point.umpbt.ber(side= "left", n.obs= 60)
```

---

type2.error.ber *Type 2 error function of the proportion test in a fixed design*

Description

For a proportion test in a fixed design, this evaluates (Type 2 error - a constant) at a specified value of the proportion.

Usage

```
type2.error.ber(alt, side = "right", null.val = 0.5, N, type1 = 0.005,
                root = 0)
```

Arguments

alt	a numeric in (0,1); the value of the proportion (consistent with side) where the Type 2 error of the fixed design test will be evaluated
side	a character; direction of the alternative hypothesis H1; has to be one of "right" or "left". **Default** is "right".
null.val	a numeric in (0,1); the hypothesized value of the proportion under a simple null hypothesis; **Default** is 0.5.
N	a positive numeric (integer); sample size to be used
type1	a numeric in (0,1); prespecified Type 1 error **Default** is 0.005.
root	a numeric; **Default** is 0, in which case the Type 2 error is returned; in general, (Type 2 error - root) is returned **Default** is 0.

Details

In case of a test for binomial proportion in a fixed design, this function evaluates the Type 2 error at a specified value alt in the composite alternative region. We can also use this function to do the other way round by using the argument root; that is, given a Type 2 error β, we can obtain the value of the proportion in the composite alternative region which has Type 2 error β. For 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, this returns a numeric (Type 2 error - k) at the speified alternative value alt.

Author(s)

Sandipan Pramanik, Valen E. Johnson and Anirban Bhattacharya

Examples

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

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

`type2.error.norm` *Type 2 error function of the Z-test in a fixed design*

Description

For a Z-test in a fixed design, this evaluates (Type 2 error - a constant) at a specified value of the population mean.

Usage

```
type2.error.norm( alt, side = "right", null.val = 0, sigma0 = 1,
                  N, type1 = 0.005, root = 0)
```

Arguments

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

Details

In case of a Z-test in a fixed design, this function evaluates the Type 2 error at a specified value `alt` in the composite alternative region. We can also use this function to do the other way round by using the argument `root`; that is, given a Type 2 error β, we can obtain the value of the proportion in the composite alternative region which has Type 2 error β. For 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`, this returns a numeric `(`Type 2 error - `k)` at the speified alternative value `alt`.

Author(s)

Sandipan Pramanik, Valen E. Johnson and Anirban Bhattacharya

Examples

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

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

---

type2.error.t *Type 2 error function of the T-test in a fixed design*

Description

For a T-test in a fixed design, this evaluates (Type 2 error - a constant) at a specified value of the population mean. This value should be consistent with the direction of the alternative hypothesis.

Usage

```
type2.error.t( alt, side = "right", null.val = 0, N, type1 = 0.005, root = 0)
```

Arguments

side	a character; direction of the alternative hypothesis H1; has to be one of "right" or "left"; **Default** is "right".
alt	a numeric; the value of the population mean (consistent with side) where the Type 2 error of the fixed design test will be evaluated
null.val	a numeric; hypothesized value of the population mean under a simple null hypothesis; **Default** is 0.
N	a positive numeric (integer); sample size to be used
type1	a numeric in (0,1); prespecified Type 1 error; **Default** is 0.005.
root	a numeric; **Default** is 0, in which case the Type 2 error is returned; in general, (Type 2 error - root) is returned; **Default** is 0.

Details

In case of a T-test in a fixed design, this function evaluates the Type 2 error at a specified value `alt` in the composite alternative region. But we can also use this function to do the other way round by using the argument `root`; that is, given a Type 2 error β, we can obtain the value of the proportion in the composite alternative region which has Type 2 error β. For 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`, this returns a numeric `(Type 2 error - k)` at the speified alternative value `alt`.

Author(s)

Sandipan Pramanik, Valen E. Johnson and Anirban Bhattacharya

Examples

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

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

`ump.match.ber` *Finding the "evidence threshold (δ)" in a proportion test*

Description

This function solves for δ by matching the rejection region from the UMPBT with that of the corresponding fixed design test. Basically this solves equation (20) of the supplemental information.

Usage

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

Arguments

`side`	a character; direction of the alternative hypothesis H1; has to be one of `"right"` or `"left"`; **Default** is `"right"`.
`type1`	a numeric in `(0,1)`; the prespecified Type 1 error; **Default** is `0.005`.
`n.obs`	a positive integer; number of samples to be used
`p0`	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 value of δ, the "evidence threshold", as defined in Johnson (2013). The value of δ is such that the rejection region from the UMPBT matches with that of the corresponding fixed design test.

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.ber(n.obs= 60, p0= .2)
```

Index