Package 'sequential.pops'

May 9, 2026

Type Package

Title Sequential Analysis of Biological Population Sizes

Version 0.1.1

Description In population management, data come at more or less regular intervals over time in sampling batches (bouts) and decisions should be made with the minimum number of samples and as quickly as possible. This package provides tools to implement, produce charts with stop lines, summarize results and assess sequential analyses that test hypotheses about population sizes. Two approaches are included: the sequential test of Bayesian posterior probabilities (Rincon, D.F. et al. 2025 <doi:10.1111/2041-210X.70053>), and the sequential probability ratio test (Wald, A. 1945 <http://www.jstor.org/stable/2235829>).

Encoding UTF-8

Imports methods, emdbook, truncdist, rlang

RoxygenNote 7.3.2

Suggests knitr, rmarkdown, testthat (>= 3.0.0)

Config/testthat/edition 3

License GPL (>= 3)

URL https://github.com/rincondf/sequential.pops, https://rincondf.github.io/sequential.pops/

BugReports https://github.com/rincondf/sequential.pops/issues

VignetteBuilder knitr

NeedsCompilation no

Author Diego F Rincon [aut, cre, cph] (ORCID: <https://orcid.org/0000-0001-6869-4430>), Izzy McCabe [aut] (ORCID: <https://orcid.org/0009-0004-5179-0345>), David W Crowder [aut] (ORCID: <https://orcid.org/0000-0002-3720-1581>)

Maintainer Diego F Rincon <diego.rincon@wsu.edu>

Repository CRAN

Date/Publication 2025-06-05 11:20:02 UTC

Contents

`int_eval` *Simulation of an STBP*

Description

This function is called on Evaluation of Sequential tests of Bayesian posterior probabilities, `STBP.eval`.

Usage

```
int_eval(pop_mean, prior, n, obj, overdispersion.sim = NA, seed = NULL)
```

Arguments

`pop_mean`	True population density.
`prior`	Initial prior.
`n`	Sample size within bouts.
`obj`	A STBP object
`overdispersion.sim`	Overdispersion parameter or function for simulations.
`seed`	Optional seed for random count generation.

Value

A list with average number of bouts required to reach a decision and recommendation for H

int_eval2 *Simulation of an SPRT*

Description

This function is called on Evaluation of Sequential probability ratio test, `SPRT.eval`.

Usage

```
int_eval2(pop_mean, obj, overdispersion.sim = NA, seed = NULL)
```

Arguments

`pop_mean`	True population density.
`obj`	A SPRT object
`overdispersion.sim`	Overdispersion parameter or function for simulations.
`seed`	Optional seed for random count generation.

Value

A list with average number of bouts required to reach a decision and recommendation for H1

plot,SPRT,missing-method *Draws a plot with the cumulative introduced counts contrasted with stop lines or stop lines when no data is provided*

Description

Method for signature `"SPRT"` to display results or stop lines.

Usage

```
## S4 method for signature 'SPRT,missing'
plot(x, y)
```

Arguments

`x`	Created as a result of a call to `sprt`.
`y`	Unused entry.

Value

When data is provided, a plot with cumulative counts contrasted with stop lines from a `"SPRT"`. When no data is provided, a plot with stop lines.

Examples

```
test00 <- sprt(mu0 = 2,
               mu1 = 4,
               density_func = "negative binomial",
               overdispersion = 4.6,
               alpha = 0.1,
               beta = 0.1)
plot(test00) # returns chart with stop lines

counts <- c(2, 5, 6, 2, 7)

test11 <- sprt(data = counts,
               mu0 = 2,
               mu1 = 4,
               density_func = "negative binomial",
               overdispersion = 4.6,
               alpha = 0.1,
               beta = 0.1)
plot(test11) # returns chart with data and stop lines

## End (Not run)
```

---

plot,STBP,missing-method

Draws a plot with the sequence of posterior probabilities from a sequential test of Bayesian posterior probabilities

---

Description

Method for signature "STBP" to display resulting probabilities.

Usage

```
## S4 method for signature 'STBP,missing'
plot(x, y)
```

Arguments

x	Created as a result of a call to stbp_simple or stbp_composite.
y	Unused entry

Value

A plot with the sequence of posterior probabilities

Examples

```
# Testing the hypothesis of a sampled population being greater than trajectory H
H <- c(2, 5, 10, 20, 40, 40, 20, 10, 5, 2)

# Generating sequential samples (n = 3) from a population that is 1 below H
# (H - 1)

countP <- matrix(NA, 3, 10)
set.seed(101)
for(i in 1:10){
  countP[, i] <- rpois(3, lambda = (H[i] - 1))
}

# Running STBP on the sample

test2F <- stbp_composite(data = countP,
                         greater_than = TRUE,
                         hypothesis = H,
                         density_func = "poisson",
                         prior = 0.5,
                         lower_bnd = 0,
                         upper_bnd = Inf,
                         lower_criterion = 0.001,
                         upper_criterion = 0.999)
plot(test2F)

## End (Not run)
```

`show,SPRT-method` *Displays results from a "SPRT" object nicely*

Description

Method for signature `"SPRT"` to show results or test specification.

Usage

```
## S4 method for signature 'SPRT'
show(object)
```

Arguments

`object`	Created as a result of a call to `sprt`.

Value

A summary of test results or specification (if data = NA or omitted).

Examples

```
test00 <- sprt(mu0 = 2,
               mu1 = 4,
               density_func = "negative binomial",
               overdispersion = 4.6,
               alpha = 0.1,
               beta = 0.1)
show(test00) # returns test specification.

counts <- c(2, 5, 6, 2, 7)

test11 <- sprt(data = counts,
               mu0 = 2,
               mu1 = 4,
               density_func = "negative binomial",
               overdispersion = 4.6,
               alpha = 0.1,
               beta = 0.1)
show(test11) # returns "accept H1" after 5 sampling bouts processed.

## End (Not run)
```

---

show,STBP-method *Displays results from a "STBP" object nicely*

Description

Method for signature "STBP" to show results.

Usage

```
## S4 method for signature 'STBP'
show(object)
```

Arguments

object	Created as a result of a call to stbp_simple or stbp_composite.

Value

A summary of the test.

Examples

```
set.seed(101)
counts3 <- rpois(5, lambda = 3)

test1F <- stbp_composite(data = counts3,
                         greater_than = TRUE,
```

```
                          hypothesis = 5,
                          density_func = "poisson",
                          prior = 0.5,
                          lower_bnd = 0,
                          upper_bnd = Inf,
                          lower_criterion = 0.001,
                          upper_criterion = 0.999)
show(test1F)
# returns "reject H".

counts10 <- matrix(rep(0, 30), 10, 3)

test1G <- stbp_simple(data = counts10,
                        density_func= "poisson",
                        prior = 0.5,
                        upper_bnd = Inf,
                        lower_criterion = 0,
                        upper_criterion = 0.9999)
show(test1G)

# returns "keep sampling" due to insufficient evidence.

## End (Not run)
```

`sprt` *Sequential Probability Ratio Test*

Description

Runs a Sequential Probability Ratio Test for hypotheses about population densities of the form $H_0 : \mu = \psi_0$ vs. $H_1 : \mu < \psi_1$, where $\psi_0 < \psi_1$. Data is treated in a sequential framework.

Usage

```
sprt(data = NA, mu0, mu1, density_func, overdispersion, alpha, beta)
```

Arguments

`data`	Optional vector of count data (NAs not allowed). Each value is considered a sampling bout over time. Can't process group sequential data. If not provided (NA), returns a `"SPRT"` object that can be used to plot a chart with stop lines or to return stop lines coefficients. If provided, returns a `"SPRT"` object that can be used to plot a chart with stop lines and data or to return test summary.
`mu0`	Single non-negative number with the value for the low hypothesized population density, ψ_0.
`mu1`	Single non-negative number with the value for the high hypothesized population density, ψ_1.
`density_func`	Kernel probability density function for the data. See details.

overdispersion	A number specifying the overdispersion parameter. Only required when using "negative binomial" as kernel density. See details.
alpha	Single number indicating tolerable type I error rate.
beta	Single number indicating tolerable type II error rate.

Details

The density_func argument should be specified as character string. Acceptable options are "poisson", "negative binomial", and "binomial". As far as we know, no one has ever calculated and published stop lines for the beta-binomial family. The overdispersion parameter should only be specified as a constant. In contrast to the STBP, SPRT is only use overdispersion to calculate stop lines, so the estimate for the threshold population density should be used (e.g., at $\psi_0 < \psi < \psi_1$).

Value

An object of class "SPRT".

References

Binns, M.R., Nyrop, J.P. & Werf, W.v.d. (2000) *Sampling and monitoring in crop protection: the theoretical basis for developing practical decision guides*. CABI Pub., Wallingford, Oxon, UK; New York, N.Y.

Wald, A. (1945) Sequential Tests of Statistical Hypotheses. *The Annals of Mathematical Statistics* 16(2): 117-186.

Examples

```
# If no data is provided, an object of class "SPRT" is returned from which a
# chart with stop lines or a summary of the test with coefficients for stop lines
# can be extracted.

test00 <- sprt(mu0 = 2,
               mu1 = 4,
               density_func = "negative binomial",
               overdispersion = 4.6,
               alpha = 0.1,
               beta = 0.1)

test00 # returns test specification and stop lines coefficients
plot(test00) # returns a chart with stop lines


# If data is provided, an object of class "SPRT" is returned with test results.

counts <- c(2, 5, 6, 2, 7)

test11 <- sprt(data = counts,
               mu0 = 2,
               mu1 = 4,
               density_func = "negative binomial",
```

```
                overdispersion = 4.6,
                alpha = 0.1,
                beta = 0.1)
test11 # returns "accept H1" after 5 sampling bouts processed.

## End (Not run)
```

SPRT-class *Class "SPRT". Result of a Sequential Probability Ratio Test*

Description

This class encapsulates results or specification of a Sequential Probability Ratio Test.

Slots

`call` (language) The call to `sprt`.

`data` (numeric, logical) Either a vector with cumulative counts or NA.

`hi_int` (numeric) Intercept of the lower stop line.

`low_int` (numeric) Intercept of the higher stop line.

`slope` (numeric) Slope of stop lines.

`recommendation` (character, logical) Either a recommendation on whether to accept H0 or H1, or keep sampling, or NA.

`iterations` (numeric) Number of sequential sampling bouts required or processed.

Examples

```
counts <- c(2, 5, 6, 2, 7)

test11 <- sprt(data = counts,
                mu0 = 2,
                mu1 = 4,
                density_func = "negative binomial",
                overdispersion = 4.6,
                alpha = 0.1,
                beta = 0.1)
show(test11) # returns "accept H1" after 5 sampling bouts processed.

## End (Not run)
```

SPRT.eval *Evaluation of Sequential probability ratio tests*

Description

Obtains the average number of sampling bouts and the rate of acceptance for H_1, from a sequential test of $H_0 : \mu = \psi_0$ vs. $H_1 : \mu < \psi_1$ across a range of true population densities, based on simulations. Sometimes called "operating characteristics".

Usage

```
SPRT.eval(obj, eval.range, overdispersion.sim = NA, N, seed = NULL)
```

Arguments

obj	An object of class "SPRT".
eval.range	A vector with a sequence of true population densities to evaluate.
overdispersion.sim	A character string (if a function) or a non-negative number specifying the overdispersion parameter used to generate simulated counts. Only required when using "negative binomial" or "beta-binomial" as kernel densities. See details.
N	Number of simulations per true population density being evaluated.
seed	Optional seed for random count generation.

Details

The kernel probability density function to evaluate the test is that specified in the argument density_func to create the "SPRT" object, but overdispersion can be different to generate simulated counts. If "negative binomial" is used as kernel density for the test and overdispersion.sim is not specified (NA), then the same specification of the test is used to generate the counts. Ideally, overdispersion for simulations should include uncertainty about the parameter to produce more robust test evaluations. For example, if using a negative binomial kernel and the Taylor's Power Law approach to obtain overdispersion, then overdispersion for simulations should be specified as:

$$k = \frac{\mu^2}{a\mu^b e^z - \mu}$$

where k is the overdispersion parameter of the negative binomial distribution, a and b are parameters of the Taylor's Power Law and z is a normally distributed variable with mean 0 and standard deviation σ_e, which is the root of the mean square error for the regression used to estimate a and b. See examples.

Value

A list with the average number of sampling bouts required to reach a decision ($AvgSamples), and the rate of acceptance for H_1 across the provided range of population densities ($AcceptRate).

References

Binns, M.R., Nyrop, J.P. & Werf, W.v.d. (2000) *Sampling and monitoring in crop protection: the theoretical basis for developing practical decision guides*. CABI Pub., Wallingford, Oxon, UK; New York, N.Y.

Examples

```
# Function that describes negative binomial overdisperion from the mean
# and Taylor's Power Law parameters, a and b:

estimate_k <- function(mean) {
                       a = 1.830012
                       b = 1.218041 # a and b are Taylor's Power Law parameters
                       (mean^2) / ((a * mean^(b)) - mean)
                      }

# Generate a SPRT object with negative binomial and varying overdispersion

counts <- c(2, 5, 6, 2, 7)

test11 <- sprt(data = counts,
               mu0 = 8,
               mu1 = 10,
               density_func = "negative binomial",
               overdispersion = estimate_k(9),
               alpha = 0.1,
               beta = 0.1)

# Stochastic version of 'estimate k', where sd here is the the root of the
# mean square error for the regression used to estimate a and b.

estimate_k_stoch <- function(mean) {
                            a <- 1.830012
                            b <- 1.218041
                            (mean^2) /
                              ((a * mean^(b) *
                                exp(truncdist::rtrunc(1,
                                                      "norm",
                                                      a = log(1 / (a * mean^(b - 1))),
                                                      b = Inf,
                                                      mean = 0,
                                                      sd = 0.3222354)))
                                      - mean)
                               }

# Run model evaluation for test11 with varying overdispersion and
# added stochasticity.

eval4 <- SPRT.eval(test11, eval.range = seq(4, 13),
                   overdispersion.sim = "estimate_k_stoch", N = 30)

plot(seq(4, 13), eval4$AvgSamples, type = "o", xlab = "True population size",
```

```
      ylab = "Average number of bouts")
plot(seq(4, 13), eval4$AcceptRate, type = "o", xlab = "True population size",
      ylab = "Acceptance rate of H1")

## End (Not run)
```

STBP-class *Class "STBP". Result of a Sequential tests of Bayesian posterior probabilities*

Description

This class encapsulates results of a Sequential tests of Bayesian posterior probabilities

Slots

`call` (language) The call to `stbp_simple` or `stbp_composite`.

`probabilities` (numeric) Vector of sequential posterior probabilities.

`recommendation` (character) Recommendation on H, whether to accept, reject or keep sampling.

`iterations` (numeric) Number of sequential sampling bouts required or processed.

Examples

```
set.seed(101)
counts3 <- rpois(5, lambda = 3)

test1F <- stbp_composite(data = counts3,
                         greater_than = TRUE,
                         hypothesis = 5,
                         density_func = "poisson",
                         prior = 0.5,
                         lower_bnd = 0,
                         upper_bnd = Inf,
                         lower_criterion = 0.001,
                         upper_criterion = 0.999)
test1F # returns "reject H".

counts10 <- matrix(rep(0, 30), 10, 3)

test1G <- stbp_simple(data = counts10,
                      density_func= "poisson",
                      prior = 0.5,
                      upper_bnd = Inf,
                      lower_criterion = 0,
                      upper_criterion = 0.9999)
test1G # returns "keep sampling" due to insufficient evidence.

## End (Not run)
```

STBP.eval *Evaluation of Sequential tests of Bayesian posterior probabilities*

Description

Obtains the average number of sampling bouts and the rate of acceptance for $H : \mu > \psi$ or $H : \mu < \psi$ across a range of true population densities, based on simulations. Sometimes called "operating characteristics".

Usage

```
STBP.eval(obj, eval.range, n, prior, overdispersion.sim = NA, N, seed = NULL)
```

Arguments

obj	An object of class "STBP".
eval.range	The evaluation range. A vector with a sequence of true population densities to evaluate. If the hypothesis argument in the test is a trajectory (i.e., a vector), the evaluation range is a factor of the hypothesized trajectory. See details.
n	Sample size within bouts.
prior	Single number with initial prior. Must be on the interval $[0, 1]$.
overdispersion.sim	A character string (if a function) or a a non-negative number specifying the overdispersion parameter used to generate simulated counts. Only required when using "negative binomial" or "beta-binomial" as kernel densities. See details.
N	Number of simulations per true population density or trajectory being evaluated.
seed	Optional seed for random count generation.

Details

The kernel probability density function to evaluate the test is that specified in the argument density_func to create the "STBP" object, but overdispersion can be different to generate simulated counts. If "negative binomial" or "beta-binomial" are used as kernel densities for the test and overdispersion.sim is not specified (NA), then the same specification of the test is used to generate the counts. Ideally, overdispersion for simulations should include uncertainty about the parameter to produce more robust test evaluations. For example, if using a negative binomial kernel and the Taylor's Power Law approach to obtain overdispersion, then overdispersion for simulations should be specified as:

$$k = \frac{\mu^2}{a\mu^b e^z - \mu}$$

where k is the overdispersion parameter of the negative binomial distribution, a and b are parameters of the Taylor's Power Law and z is a normally distributed variable with mean 0 and standard deviation σ_e, which is the root of the mean square error for the regression used to estimate a and b. See examples.

The evaluation range in the `eval.range` argument should cover relevant population densities for which the test will be applied. These densities are often around the threshold to check sampling sizes and error rates at those critical levels. In the case of dynamic hypotheses (vectors), the range should be given as factors of the hypothesized trajectory. For example, to evaluate this test for a hypothesis trajectory the `eval.range` argument could be `seq(0.1, 2, 0.1)`, so the evaluation runs from 10 percent of the trajectory to twice the trajectory in 10 percent increments.

Value

A list with the average number of sampling bouts required to reach a decision (`$AvgSamples`), and the rate of acceptance for H across the provided range of population densities (`$AcceptRate`).

References

Binns, M.R., Nyrop, J.P. & Werf, W.v.d. (2000) *Sampling and monitoring in crop protection: the theoretical basis for developing practical decision guides*. CABI Pub., Wallingford, Oxon, UK; New York, N.Y.

Rincon, D.F., McCabe, I. & Crowder, D.W. (2025) Sequential testing of complementary hypotheses about population density. *Methods in Ecology and Evolution*. <https://doi.org/10.1111/2041-210X.70053>

Examples

```
# These examples are run with very few simulation runs (argument N), so they
# provide unrealistic results. For more reasonable demonstrations check the
# vignettes.

# Assuming a negative binomial count variable whose overdispersion parameter,
# k, varies as a function of the mean, and that the variance-mean relationship
# is well described with Taylor's Power Law, a function to obtain k can be:

estimate_k <- function(mean) {
                        a = 1.830012
                        b = 1.218041 # a and b are Taylor's Power Law parameters
                        (mean^2) / ((a * mean^(b)) - mean)
                        }

# Generate some counts to create an STBP object with the model specifications

counts3 <- rnbinom(20, mu = 5, size = estimate_k(5))

# Run the test to create the STBP object

test1F <- stbp_composite(data = counts3,
                          greater_than = TRUE,
                          hypothesis = 9,
                          density_func = "negative binomial",
                          overdispersion = "estimate_k",
                          prior = 0.5,
                          lower_bnd = 0,
                          upper_bnd = Inf,
```

```
                          lower_criterion = 0.01,
                          upper_criterion = 0.99)

test1F

# Model evaluation is carried out based on simulated counts, and more realistic
# counts could be generated if uncertainty about the overdispersion parameter is
# considered. A function to obtain values for the overdispersion parameter, k,
# with added stochasticity could be (following Binns et al. 2000):

estimate_k_stoch <- function(mean) {
                              a <- 1.830012
                              b <- 1.218041
                              (mean^2) /
                                ((a * mean^(b) *
                                  exp(truncdist::rtrunc(1,
                                                        "norm",
                                                        a = log(1 / (a * mean^(b - 1))),
                                                        b = Inf,
                                                        mean = 0,
                                                        sd = 0.3222354)))
                                       - mean)
                                 }

# where sd here is the the root of the mean square error for the regression
# used to estimate a and b. Note that this is a stochastic version
# of 'estimate_k'.

# Run model evaluation for testF1 with varying overdispersion and
# added stochasticity with very few simulations to save time.

eval1 <- STBP.eval(test1F,
                   eval.range = seq(2, 11),
                   n = 1, prior = 0.5,
                   overdispersion.sim = "estimate_k_stoch",
                   N = 2)

plot(seq(2, 11), eval1$AvgSamples, type = "o", xlab = "True population size",
      ylab = "Average number of bouts")

plot(seq(2, 11), eval1$AcceptRate, type = "o", xlab = "True population size",
      ylab = "Acceptance rate of H")

# Alternatively, the evaluation could be carried out omitting variation about
# overdispersion. For that the overdispersion argument is omitted and the same
# specification of the model is used. Very few simulations to save time.

eval2 <- STBP.eval(test1F,
                   eval.range = seq(2, 11),
                   n = 1, prior = 0.5,
                   N = 2)

plot(seq(2, 11), eval2$AvgSamples, type = "o", xlab = "True population size",
```

```
      ylab = "Average number of bouts")

plot(seq(2, 11), eval2$AcceptRate, type = "o", xlab = "True population size",
      ylab = "Acceptance rate of H")

# When there is no overdispersion (poisson or binomial distributions) the
# procedure is much simpler

test2F <- stbp_composite(data = counts3,
                         greater_than = TRUE,
                         hypothesis = 5,
                         density_func = "poisson",
                         prior = 0.5,
                         lower_bnd = 0,
                         upper_bnd = Inf,
                         lower_criterion = 0.01,
                         upper_criterion = 0.99)

test2F

# Overdispersion is omitted here. Again, very few simulations (N) to save time.

eval3 <- STBP.eval(test2F,
                   eval.range = seq(1, 8),
                   n = 1,
                   prior = 0.5, N = 2)

plot(seq(1, 8), eval3$AvgSamples, type = "o", xlab = "True population size",
      ylab = "Average number of bouts")

plot(seq(1, 8), eval3$AcceptRate, type = "o", xlab = "True population size",
      ylab = "Acceptance rate of H")

# Variations if n, the sample size within each bout, can also be changed
# (not possible in SPRT)!

## End (Not run)
```

`stbp_composite` *Sequential test of Bayesian posterior probabilities for composite hypotheses*

Description

Runs a Sequential test of Bayesian Posterior Probabilities for hypotheses about population densities of the form $H : \mu > \psi$ or $H : \mu < \psi$. Data is treated in a sequential framework.

Usage

```
stbp_composite(
```

```
  data,
  greater_than = TRUE,
  hypothesis,
  density_func,
  overdispersion = NA,
  prior = 0.5,
  lower_bnd = 0,
  upper_bnd = Inf,
  lower_criterion,
  upper_criterion
)
```

Arguments

`data`	For count data, either a vector (for purely sequential designs) or a matrix (group sequential designs) with sequential (non-negative) count data, with sampling bouts collected over time in columns and samples within bouts in rows. NAs are allowed in case sample size within bouts is unbalanced. For binomial data, a list of matrices with integer non-negative values of observations in col 1 and number of samples in col 2, so that each matrix within the list corresponds to a sampling bout. NAs are *not* allowed for binomial data.
`greater_than`	logical; if TRUE (default), the tested hypothesis is of the form $H : \mu > \psi$ otherwise, $H : \mu < \psi$.
`hypothesis`	Either a single non-negative value or a vector of non-negative values with the hypothesized population densities, ψ. If a vector, it should contain at least as many values as `ncol(data)` for count data or as `length(data)` for binomial data.
`density_func`	Kernel probability density function for the data. See details.
`overdispersion`	A character string (if a function) or a number specifying the overdispersion parameter. Only required when using `"negative binomial"` or `"beta-binomial"` as kernel densities. See details.
`prior`	Single number with initial prior. Must be on the interval $[0, 1]$. If no prior information is available 0.5 (default) is recommended.
`lower_bnd`	Single number indicating the lower bound of the parameter space for μ. Most cases is 0 (default).
`upper_bnd`	Single number indicating the upper bound of the parameter space for μ. For count data, is often `Inf` (default), but it must be ≤ 1 for binomial data.
`lower_criterion`	Criterion to decide against the tested hypothesis. This is the maximum credibility to the hypothesis to stop sampling and decide against.
`upper_criterion`	Criterion to decide in favor of the tested hypothesis. This is the minimum credibility to the hypothesis to stop sampling and decide in favor.

Details

The density_func argument should be specified as character string. Acceptable options are "poisson", "negative binomial", "binomial" and "beta-binomial". The overdispersion parameter for "negative binomial" and "beta-binomial" can be either a constant or a function of the mean.

If a function, it should be specified as a character string with the name of an existing function. For options of empirical functions to describe overdispersion as a function of the mean see Binns et al. (2000). The most common approach for the negative binomial family is Taylor's Power Law, which describes the variance as a function of the mean with two parameters, a and b. Overdispersion, k, can then be specified as:

$$k = \frac{\mu^2}{a\mu^b - \mu}$$

Value

An object of class "STBP".

References

Binns, M.R., Nyrop, J.P. & Werf, W.v.d. (2000) *Sampling and monitoring in crop protection: the theoretical basis for developing practical decision guides*. CABI Pub., Wallingford, Oxon, UK; New York, N.Y.

Rincon, D.F., McCabe, I. & Crowder, D.W. (2025) Sequential testing of complementary hypotheses about population density. *Methods in Ecology and Evolution*. <https://doi.org/10.1111/2041-210X.70053>

Examples

```
# Testing the hypothesis of a population size being greater than 5 individuals
# per sampling unit (H: mu > 5). The sequential sampling is made of 5 sampling
# bouts made of one sample each.

set.seed(101)
counts3 <- rpois(5, lambda = 3)

test1F <- stbp_composite(data = counts3,
                         greater_than = TRUE,
                         hypothesis = 5,
                         density_func = "poisson",
                         prior = 0.5,
                         lower_bnd = 0,
                         upper_bnd = Inf,
                         lower_criterion = 0.001,
                         upper_criterion = 0.999)
test1F

# returns "reject H".

# Testing the hypothesis of a sampled population being greater than trajectory H
H <- c(2, 5, 10, 20, 40, 40, 20, 10, 5, 2)
```

```
# Generating sequential samples (n = 3) from a population that is 1 below H
# (H - 1)

countP <- matrix(NA, 3, 10)
set.seed(101)
for(i in 1:10){
  countP[, i] <- rpois(3, lambda = (H[i] - 1))
}

# Running STBP on the sample

test2F <- stbp_composite(data = countP,
                         greater_than = TRUE,
                         hypothesis = H,
                         density_func = "poisson",
                         prior = 0.5,
                         lower_bnd = 0,
                         upper_bnd = Inf,
                         lower_criterion = 0.001,
                         upper_criterion = 0.999)
test2F

# returns "reject H".

# Testing the hypothesis of a proportion of infested units being greater than
# 20% per sampling unit (H: mu > 0.2). The sequential sampling is made of 7
# sampling bouts each with 5 clusters of 10 samples from which binomial
# observations are recorded.

set.seed(101)

# binomial data generated with mu (prob) 0.05 over the hypothesized
# value (0.2)

counts4 <- list()
for(i in 1: 7) {
  counts4[[i]] <- matrix(c(rbinom(5, size = 10, prob = 0.25), rep(10, 5)),
                         5, 2)
}

# Run the test. Notice that upper_bnd = 1!

test3F <- stbp_composite(data = counts4,
                         greater_than = TRUE,
                         hypothesis = 0.2,
                         density_func = "binomial",
                         prior = 0.5,
                         lower_bnd = 0,
                         upper_bnd = 1,
                         lower_criterion = 0.001,
                         upper_criterion = 0.999)

test3F # returns accept H after 3 sampling bouts
```

```
# Assuming a negative binomial count variable whose overdispersion parameter,
# k, varies as a function of the mean, and that the variance-mean relationship
# is well described with Taylor's Power Law, a function to obtain k can be:

estimate_k <- function(mean) {
                        a = 1.830012
                        b = 1.218041 # a and b are Taylor's Power Law parameters
                        (mean^2) / ((a * mean^(b)) - mean)
                        }

# Generate some counts to create an STBP object with the model specifications

counts3 <- rnbinom(20, mu = 5, size = estimate_k(5))

# Run the test to create the STBP object

test1F <- stbp_composite(data = counts3,
                          greater_than = TRUE,
                          hypothesis = 9,
                          density_func = "negative binomial",
                          overdispersion = "estimate_k",
                          prior = 0.5,
                          lower_bnd = 0,
                          upper_bnd = Inf,
                          lower_criterion = 0.01,
                          upper_criterion = 0.99)

test1F

## End (Not run)
```

`stbp_posterior_composite` *Posterior probability calculation for composite hypotheses*

Description

This function calculates a posterior probability for hypotheses about population densities of the form $H : \mu > \psi$ or $H : \mu < \psi$, given the data at a single iteration. This function is to be used in a sequential framework, and called on the sequential test `stbp_composite`.

Usage

```
stbp_posterior_composite(
  data,
  greater_than,
  hypothesis,
  density_func,
```

```
  overdispersion = NA,
  prior,
  lower_bnd = 0,
  upper_bnd = Inf
)
```

Arguments

data	For count data, a numeric vector with for a single sampling bout (NAs allowed). For binomial data, a matrix with observations in col 1 and samples in col 2 (NAs *not* allowed).
greater_than	logical; if TRUE, the tested hypothesis is of the form $H : \mu > \psi$ otherwise, $H : \mu < \psi$.
hypothesis	Single non-negative value with the hypothesized value of μ.
density_func	Kernel probability density function for the data. See details.
overdispersion	A character string (if a function) or a number specifying the overdispersion parameter. Only required when using "negative binomial" or "beta-binomial" as kernel densities. See details.
prior	Single number with initial prior. Must be on the interval $[0, 1]$.
lower_bnd	Single number indicating the lower bound of the parameter space for μ. Most cases is 0 (default).
upper_bnd	Single number indicating the upper bound of the parameter space for μ. For count data, is often Inf (default), but it must be ≤ 1 for binomial data.

Details

The density_func argument should be specified as character string. Acceptable options are "poisson", "negative binomial", "binomial" and "beta-binomial". The overdispersion parameter for "negative binomial" and "beta-binomial" can be either a constant or a function of the mean. If a function, it should be specified as a character string with the name of an existing function. For options of empirical functions to describe overdispersion as a function of the mean see Binns et al. (2000). The most common approach for the negative binomial family is Taylor's Power Law.

Value

A single probability

References

Binns, M.R., Nyrop, J.P. & Werf, W.v.d. (2000) *Sampling and monitoring in crop protection: the theoretical basis for developing practical decision guides*. CABI Pub., Wallingford, Oxon, UK; New York, N.Y.

Rincon, D.F., McCabe, I. & Crowder, D.W. (2025) Sequential testing of complementary hypotheses about population density. *Methods in Ecology and Evolution*. <https://doi.org/10.1111/2041-210X.70053>

Examples

```
# Counts collected in a single sampling bout
counts <- c(1, 2, 3)

# Calculate posterior probability from a naive 0.5 prior for H1:mu>2
# (a population being >2 individuals per sampling unit) with
# a poisson kernel

stbp_posterior_composite(data = counts,
                         greater_than = TRUE,
                         hypothesis = 2,
                         density_func = "poisson",
                         prior = 0.5,
                         lower_bnd = 0,
                         upper_bnd = Inf) # returns 0.60630278

# Same analysis but with a negative binomial kernel.
# Note that 'overdispersion' can either be a positive number or a function.

stbp_posterior_composite(data = counts,
                         greater_than = TRUE,
                         hypothesis = 2,
                         density_func = "negative binomial",
                         overdispersion = 2,
                         prior = 0.5,
                         lower_bnd = 0,
                         upper_bnd = Inf) # returns 0.72558593
## End (Not run)
```

---

stbp_posterior_simple *Posterior calculation for simple hypotheses about species absence*

Description

This function calculates a posterior probability for hypotheses about population densities, of the form $H : \mu = 0$, given the data at a single iteration. This function is to be used in a sequential framework, and called on the sequential test `stbp_simple`.

Usage

```
stbp_posterior_simple(
  data,
  density_func,
  overdispersion = NA,
  prior,
  upper_bnd = Inf
)
```

Arguments

`data`	For count data, a numeric vector with for a single sampling bout (NAs allowed). For binomial data, a matrix with observations in col 1 and samples in col 2 (NAs *not* allowed).
`density_func`	Kernel probability density function for the data. See details.
`overdispersion`	A character string (if a function) or a number specifying the overdispersion parameter. Only required when using `"negative binomial"` or `"beta-binomial"` as kernel densities. See details.
`prior`	Single number with initial prior. Must be in the interval $[0, 1]$.
`upper_bnd`	Single number indicating the greatest possible value for μ. For count data, is often `Inf` (default), but it must be ≤ 1 for binomial data.

Details

The `density_func` argument should be specified as character string. Acceptable options are `"poisson"`, `"negative binomial"`, `"binomial"` and `"beta-binomial"`. The overdispersion parameter for `"negative binomial"` and `"beta-binomial"` can be either a constant or a function of the mean. If a function, it should be specified as a character string with the name of an existing function. For options of empirical functions to describe overdispersion as a function of the mean see Binns et al. (2000). The most common approach for the negative binomial family is Taylor's Power Law.

Value

A single probability

References

Binns, M.R., Nyrop, J.P. & Werf, W.v.d. (2000) *Sampling and monitoring in crop protection: the theoretical basis for developing practical decision guides*. CABI Pub., Wallingford, Oxon, UK; New York, N.Y.

Rincon, D.F., McCabe, I. & Crowder, D.W. (2025) Sequential testing of complementary hypotheses about population density. *Methods in Ecology and Evolution*. <https://doi.org/10.1111/2041-210X.70053>

Examples

```
# Counts collected in a single sampling bout
counts <- c(0, 0, 0)

# Calculate posterior probability from a naive 0.5 prior for H:mu=0
# (a species being absent in an area) with a poisson kernel.

stbp_posterior_simple(data = counts,
                      density_func = "poisson",
                      prior = 0.5,
                      upper_bnd = Inf) # returns 0.75
## End (Not run)
```

stbp_simple *Sequential test of Bayesian posterior probabilities for simple hypotheses about species absence*

Description

Runs a Sequential test of Bayesian Posterior Probabilities for hypotheses about species absence of the form $H : \mu = 0$. Data is treated in a sequential framework.

Usage

```
stbp_simple(
  data,
  density_func,
  overdispersion = NA,
  prior = 0.5,
  upper_bnd = Inf,
  lower_criterion,
  upper_criterion
)
```

Arguments

data	For count data, either a vector (for purely sequential designs) o a matrix (group sequential designs) with sequential count data, with sampling bouts collected over time in columns and samples within bouts in rows. NAs are allowed in case sample size within bouts is unbalanced. For binomial data, a list of matrices with integer non-negative values of observations in col 1 and number of samples in col 2, so that each matrix within the list corresponds to a sampling bout. NAs are *not* allowed for binomial data.
density_func	Kernel probability density function for the data. See details.
overdispersion	A character string (if a function) or a number specifying the overdispersion parameter. Only required when using "negative binomial" or "beta-binomial" as kernel densities. See details.
prior	Single number with initial prior. Must be in the interval $[0, 1]$. If no prior information is available 0.5 (default) is recommended.
upper_bnd	Single number indicating the greatest possible value for μ. For count data, is often Inf (default), but it must be ≤ 1 for binomial data.
lower_criterion	Criterion to decide against the tested hypothesis. This is the lowest credibility to the hypothesis to stop sampling and decide against.
upper_criterion	Criterion to decide in favor of the tested hypothesis. This is the greatest credibility to the hypothesis to stop sampling and decide in favor.

Details

The density_func argument should be specified as character string. Acceptable options are "poisson", "negative binomial", "binomial" and "beta-binomial". The overdispersion parameter for "negative binomial" and "beta-binomial" can be either a constant or a function of the mean.

If a function, it should be specified as a character string with the name of an existing function. For options of empirical functions to describe overdispersion as a function of the mean see Binns et al. (2000). The most common approach for the negative binomial family is Taylor's Power Law, which describes the variance as a function of the mean with two parameters, a and b. Overdispersion, k, can then be specified as:

$$k = \frac{\mu^2}{a\mu^b - \mu}$$

Value

An object of class "STBP".

References

Binns, M.R., Nyrop, J.P. & Werf, W.v.d. (2000) *Sampling and monitoring in crop protection: the theoretical basis for developing practical decision guides*. CABI Pub., Wallingford, Oxon, UK; New York, N.Y.

Rincon, D.F., McCabe, I. & Crowder, D.W. (2025) Sequential testing of complementary hypotheses about population density. *Methods in Ecology and Evolution*. <https://doi.org/10.1111/2041-210X.70053>

Examples

```
# Testing the absence of a species in a given area from a sequential random
# sampling of 3 bouts made of 10 samples (counts) each (all absences). Upper
# criterion set to 0.9999

counts10 <- matrix(rep(0, 30), 10, 3)

test1G <- stbp_simple(data = counts10,
                      density_func = "poisson",
                      prior = 0.5,
                      upper_bnd = Inf,
                      lower_criterion = 0,
                      upper_criterion = 0.9999)
test1G

# returns a recommendation of "keep sampling" due to insufficient evidence.

# Testing the same hypothesis with the same upper criterion but from a
# sequential random sampling of 3 bouts made of 30 samples (counts) each
# (all absences).

counts30 <- matrix(rep(0, 90), 30, 3)

test2G <- stbp_simple(data = counts30,
```

```
                        density_func= "poisson",
                        prior = 0.5,
                        upper_bnd = Inf,
                        lower_criterion = 0,
                        upper_criterion = 0.9999)
test2G

# returns a recommendation of "accept H" of the species being absent from
# that area.

## End (Not run)
```

Index