Package 'depower'

May 8, 2026

Title Power Analysis for Differential Expression Studies

Version 2026.1.30

Description Provides a convenient framework to simulate, test, power, and visualize data for differential expression studies with lognormal or negative binomial outcomes. Supported designs are two-sample comparisons of independent or dependent outcomes. Power may be summarized in the context of controlling the per-family error rate or family-wise error rate. Negative binomial methods are described in Yu, Fernandez, and Brock (2017) <doi:10.1186/s12859-017-1648-2> and Yu, Fernandez, and Brock (2020) <doi:10.1186/s12859-020-3541-7>.

URL https://brettklamer.com/work/depower/,
https://bitbucket.org/bklamer/depower/

License MIT + file LICENSE

Depends R (>= 4.2.0)

Imports Rdpack, stats, mvnfast, glmmTMB, dplyr, multidplyr, ggplot2, scales

Suggests tinytest, rmarkdown

RdMacros Rdpack

Language en-US

Encoding UTF-8

RoxygenNote 7.3.3

NeedsCompilation no

Author Brett Klamer [aut, cre] (ORCID: <https://orcid.org/0000-0002-8334-4831>), Lianbo Yu [aut] (ORCID: <https://orcid.org/0000-0002-2025-2585>)

Maintainer Brett Klamer <code@brettklamer.com>

Repository CRAN

Date/Publication 2026-01-30 18:50:02 UTC

Contents

`add_power_ci` *Add confidence intervals for power estimates*

Description

Calculates and adds confidence intervals for power estimates to objects returned by `power()`. The confidence interval quantifies uncertainty about the true power parameter.

Usage

```
add_power_ci(x, ci_level = 0.95, method = c("wilson", "exact"))
```

Arguments

`x`	(data.frame) A data frame returned by `power()`, containing columns `power` and `nsims`.
`ci_level`	(Scalar numeric: `0.95`; `(0,1)`) The confidence interval level.
`method`	(Scalar character: `"wilson"`; `c("wilson", "exact")`) Method for computing confidence intervals. One of `"wilson"` (default) or `"exact"`.

Details

Power estimation via simulation is a binomial proportion problem. The confidence interval answers: "What is the plausible range of true power values given my simulation results?"

Let π denote the true power value, $\hat{\pi} = x/n$ denote the observed power value, n denote the number of simulations, and $x = \text{round}(\hat{\pi} \cdot n)$ denote the number of rejections. Two methods are available.

Wilson Score Interval:

The Wilson score interval is derived from inverting the score test. Starting with the inequality

$$\left| \frac{\hat{\pi} - \pi}{\sqrt{\pi(1-\pi)/n}} \right| \leq z_{1-\alpha/2},$$

and solving the resulting quadratic for π yields

$$\frac{\hat{\pi} + \frac{z^2}{2n} \pm z\sqrt{\frac{\hat{\pi}(1-\hat{\pi})}{n} + \frac{z^2}{4n^2}}}{1 + \frac{z^2}{n}},$$

with $z = z_{1-\alpha/2}$ and $\hat{\pi} = x/n$.

Clopper-Pearson Interval:

The Clopper-Pearson exact interval inverts the binomial test via Beta quantiles. The bounds (π_L, π_U) satisfy:

$$P(X \geq x \mid \pi = \pi_L) = \alpha/2$$

$$P(X \leq x \mid \pi = \pi_U) = \alpha/2$$

With x successes in n trials,

$$\pi_L = B^{-1}\left(\frac{\alpha}{2}; x, n - x + 1\right)$$

$$\pi_U = B^{-1}\left(1 - \frac{\alpha}{2}; x + 1, n - x\right)$$

where $B^{-1}(q; a, b)$ is the q-th quantile of Beta(a, b).

This method guarantees at least nominal coverage but is conservative (intervals are wider than necessary).

Approximate parametric tests:

When power is computed using approximate parametric tests (see `simulated()`), the power estimate and confidence/prediction intervals apply to the Monte Carlo test power $\mu_K = P(\hat{p} \leq \alpha)$ rather than the exact test power $\pi = P(p \leq \alpha)$. These quantities converge as the number of datasets simulated under the null hypothesis K increases. The minimum observable p-value is $1/(K+1)$, so $K > 1/\alpha - 1$ is required to observe any rejections. For practical accuracy, we recommend choosing $\max(5000, K \gg 1/\alpha - 1)$ for most scenarios. For example, if $\alpha = 0.05$, use `simulated(nsims = 5000)`.

Value

The input data frame with additional columns:

Name	Description
power_ci_lower	Lower bound of confidence interval.
power_ci_upper	Upper bound of confidence interval.

and added attribute "ci_info" containing the method description, method name and confidence level.

References

Newcombe RG (1998). "Two-sided confidence intervals for the single proportion: comparison of seven methods." *Statistics in Medicine*, **17**(8), 857–872. ISSN 0277-6715, 1097-0258, doi:10.1002/(SICI)10970258(19980430)17:8<857::AIDSIM777>3.0.CO;2E.,

Wilson EB (1927). "Probable Inference, the Law of Succession, and Statistical Inference." *Journal of the American Statistical Association*, **22**(158), 209–212. ISSN 0162-1459, 1537-274X, doi:10.1080/01621459.1927.10502953.,

Clopper CJ, Pearson ES (1934). "THE USE OF CONFIDENCE OR FIDUCIAL LIMITS ILLUSTRATED IN THE CASE OF THE BINOMIAL." *Biometrika*, **26**(4), 404–413. ISSN 0006-3444, 1464-3510, doi:10.1093/biomet/26.4.404.

See Also

power(), eval_power_ci(), add_power_pi()

Examples

```
#------------------------------------------------------------------------
# add_power_ci() examples
#------------------------------------------------------------------------
library(depower)

set.seed(1234)
x <- sim_nb(
  n1 = 10,
  mean1 = 10,
  ratio = c(1.4, 1.6),
  dispersion1 = 2,
  nsims = 200
) |>
  power(wald_test_nb())

# Compare methods
add_power_ci(x, method = "wilson")
add_power_ci(x, method = "exact")

# 99% confidence interval
add_power_ci(x, ci_level = 0.99)
```

```
# Plot with shaded region for confidence interval of the power estimate.
add_power_ci(x) |>
  plot()
```

`add_power_pi` *Add Bayesian posterior predictive intervals for power estimates*

Description

Calculates and adds Bayesian posterior predictive intervals for power estimates in objects returned by `power()`. The posterior predictive interval quantifies the expected range of power estimates from a future simulation study.

Usage

```
add_power_pi(x, future_nsims = NULL, pi_level = 0.95, prior = c(1, 1))
```

Arguments

`x`	(data.frame) A data frame returned by `power()`, containing columns `power` and `nsims`.
`future_nsims`	(Scalar integer or `NULL`: `NULL`; `[2, Inf)`) Number of simulated data sets in the future power estimate study. If `NULL` (default), uses the same number as the original study (`nsims`).
`pi_level`	(Scalar numeric: `0.95`; `(0,1)`) The posterior predictive interval level.
`prior`	(Numeric vector of length 2: `c(1, 1)`; each `(0, Inf)`) Parameters (α, β) for the Beta prior on true power. Default `c(1, 1)` is the uniform prior. Use `c(0.5, 0.5)` for the Jeffreys prior.

Details

Power estimation via simulation is a binomial proportion problem. The posterior predictive interval answers: "If I run a new simulation study with m simulations, what range of power estimates might I observe?"

Let π denote the true power value, $\hat{\pi} = x/n$ denote the observed power value, n denote the number of simulations, and $x = \text{round}(\hat{\pi} \cdot n)$ denote the number of rejections. With a Beta(α, β) prior on the true power π, the posterior after observing x successes in n trials is:

$$\pi \mid X = x \sim \text{Beta}(\alpha + x, \beta + n - x).$$

The posterior predictive distribution for Y, the number of successes in a future study with m trials, is Beta-Binomial:

$$Y \mid X = x \sim \text{BetaBinomial}(m, \alpha + x, \beta + n - x).$$

The posterior predictive interval is constructed from quantiles of this distribution, expressed as proportions Y/m.

The posterior predictive mean and variance of $\hat{\pi}_{\text{new}} = Y/m$ are:

$$E[\hat{\pi}_{\text{new}} \mid X = x] = \frac{\alpha + x}{\alpha + \beta + n}$$
$$\text{Var}[\hat{\pi}_{\text{new}} \mid X = x] = \frac{(\alpha + x)(\beta + n - x)(\alpha + \beta + n + m)}{m(\alpha + \beta + n)^2(\alpha + \beta + n + 1)}.$$

Argument `future_nsims`:

The argument `future_nsims` allows you to estimate prediction interval bounds for a hypothetical future study with different number of simulations. Note that a small initial number for `nsims` results in substantial uncertainty about the true power. A correspondingly large number of future simulations `future_nsims` will more precisely estimate the true power, but the past large uncertainty is still carried forward. Therefore you still need an adequate number of simulations `nsims` in the original study, not just more in the replication `future_nsims`, to ensure narrow prediction intervals.

Approximate parametric tests:

When power is computed using approximate parametric tests (see `simulated()`), the power estimate and confidence/prediction intervals apply to the Monte Carlo test power $\mu_K = P(\hat{p} \leq \alpha)$ rather than the exact test power $\pi = P(p \leq \alpha)$. These quantities converge as the number of datasets simulated under the null hypothesis K increases. The minimum observable p-value is $1/(K+1)$, so $K > 1/\alpha - 1$ is required to observe any rejections. For practical accuracy, we recommend choosing $\max(5000, K \gg 1/\alpha - 1)$ for most scenarios. For example, if $\alpha = 0.05$, use `simulated(nsims = 5000)`.

Value

The input data frame with additional columns:

Name	Description
`power_pi_mean`	Predictive mean of future power estimate.
`power_pi_lower`	Lower bound of posterior predictive interval.
`power_pi_upper`	Upper bound of posterior predictive interval.

and added attribute `"pi_info"` containing the method description, method name, level, prior values, and future simulation count.

References

Gelman A, Carlin JB, Stern HS, Dunson DB, Vehtari A, Rubin DB (2013). *Bayesian data analysis*, Texts in statistical science series, Third edition edition. CRC Press, Taylor & Francis Group. ISBN 9781439840955.

See Also

`power()`, `eval_power_pi()`, `add_power_ci()`

Examples

```
#------------------------------------------------------------------------------
# add_power_pi() examples
#------------------------------------------------------------------------------
library(depower)

set.seed(1234)
x <- sim_nb(
  n1 = 10,
  mean1 = 10,
  ratio = c(1.4, 1.6),
  dispersion1 = 2,
  nsims = 200
) |>
  power(wald_test_nb())

# Add posterior predictive intervals
# default: predict for same number of simulations
add_power_pi(x)

# Compare posterior predictive interval width across different future
# study sizes
add_power_pi(x, future_nsims = 100)  # wider
add_power_pi(x, future_nsims = 1000) # narrower

# Use Jeffreys prior instead of uniform
add_power_pi(x, prior = c(0.5, 0.5))

# Plot with shaded region for prediction interval of the power estimate.
add_power_pi(x) |>
  plot()
```

`distribution` *Test statistic distribution under the null*

Description

Constructs a list which defines the test statistic reference distribution under the null hypothesis.

Usage

```
asymptotic()

simulated(method = "approximate", nsims = 1000L, ncores = 1L, ...)
```

Arguments

method	(Scalar string: "approximate") The method used to derive the distribution of the test statistic under the null hypothesis. Must be one of "approximate" (default) or "exact". See 'Details' for additional information.
nsims	(Scalar integer: 1000L; [2, Inf)) The number of resamples for method = "approximate". Not used for method = "exact", except for the case when the number of exact resamples exceeds approximately 1e6 and then method = "approximate" will be used as a fallback. In the power() context, nsims defines the number of simulated datasets under the null hypothesis. For this case you would typically set nsims as greater than or equal to the number of simulated datasets in the design row of the power analysis. See 'Details' for additional information.
ncores	(Scalar integer: 1L; [1, Inf)) The number of cores (number of worker processes) to use. Do not set greater than the value returned by parallel::detectCores().
...	Optional arguments for internal use.

Details

The default asymptotic test is performed for distribution = asymptotic().

When setting argument distribution = simulated(method = "exact"), the exact randomization test is defined by:

- Independent two-sample tests
  1. Calculate the observed test statistic.
  2. Check if length(combn(x=n1+n2, m=n1))<1e6
     (a) If TRUE continue with the exact randomization test.
     (b) If FALSE revert to the approximate randomization test.
  3. For all combn(x=n1+n2, m=n1) permutations:
     (a) Assign corresponding group labels.
     (b) Calculate the test statistic.
  4. Calculate the exact randomization test p-value as the mean of the logical vector resampled_test_stats >= observed_test_stat.
- Dependent two-sample tests
  1. Calculate the observed test statistic.
  2. Check if npairs < 21 (maximum 2^20 resamples)
     (a) If TRUE continue with the exact randomization test.
     (b) If FALSE revert to the approximate randomization test.
  3. For all 2^npairs permutations:
     (a) Assign corresponding pair labels.
     (b) Calculate the test statistic.
  4. Calculate the exact randomization test p-value as the mean of the logical vector resampled_test_stats >= observed_test_stat.

For argument `distribution = simulated(method = "approximate")`, the approximate randomization test is defined by:

- Independent two-sample tests
  1. Calculate the observed test statistic.
  2. For `nsims` iterations:
     (a) Randomly assign group labels.
     (b) Calculate the test statistic.
  3. Insert the observed test statistic to the vector of resampled test statistics.
  4. Calculate the approximate randomization test p-value as the mean of the logical vector `resampled_test_stats >= observed_test_stat`.
- Dependent two-sample tests
  1. Calculate the observed test statistic.
  2. For `nsims` iterations:
     (a) Randomly assign pair labels.
     (b) Calculate the test statistic.
  3. Insert the observed test statistic to the vector of resampled test statistics.
  4. Calculate the approximate randomization test p-value as the mean of the logical vector `resampled_test_stats >= observed_test_stat`.

In the power analysis setting, `power()`, we can simulate data for groups 1 and 2 using their known distributions under the assumptions of the null hypothesis. Unlike above where nonparametric randomization tests are performed, in this setting approximate parametric tests are performed.

For example, `power(wald_test_nb(distribution = simulated()))` would result in an approximate parametric Wald test defined by:

1. For each relevant design row in `data`:
   (a) For `simulated(nsims=integer())` iterations:
      i. Simulate new data for group 1 and group 2 under the null hypothesis.
      ii. Calculate the Wald test statistic, χ^2_{null}.
   (b) Collect all χ^2_{null} into a vector.
   (c) For each of the `sim_nb(nsims=integer())` simulated datasets:
      i. Calculate the Wald test statistic, χ^2_{obs}.
      ii. Calculate the p-value based on the empirical null distribution of test statistics, χ^2_{null}. (the mean of the logical vector `null_test_stats >= observed_test_stat`)
   (d) Collect all p-values into a vector.
   (e) Calculate power as `sum(p <= alpha) / nsims`.
2. Return all results from `power()`.

Randomization tests use the positive-biased p-value estimate in the style of Davison and Hinkley (1997) (see also Phipson and Smyth (2010)):

$$\hat{p} = \frac{1 + \sum_{i=1}^{B} \mathbb{I}\{\chi_i^2 \geq \chi_{obs}^2\}}{B + 1}.$$

The number of resamples defines the minimum observable p-value (e.g. `nsims=1000L` results in min(p-value)=1/1001). It's recommended to set nsims $\gg \frac{1}{\alpha}$.

Value

list

References

Davison AC, Hinkley DV (1997). *Bootstrap Methods and their Application*, 1 edition. Cambridge University Press. ISBN 9780521574716, doi:10.1017/CBO9780511802843.

Phipson B, Smyth GK (2010). "Permutation P-values Should Never Be Zero: Calculating Exact P-values When Permutations Are Randomly Drawn." *Statistical Applications in Genetics and Molecular Biology*, **9**(1). ISSN 1544-6115, doi:10.48550/arXiv.1603.05766.

Examples

```
#-------------------------------------------------------------------------
# asymptotic() examples
#-------------------------------------------------------------------------
library(depower)

set.seed(1234)
data <- sim_nb(
  n1 = 60,
  n2 = 40,
  mean1 = 10,
  ratio = 1.5,
  dispersion1 = 2,
  dispersion2 = 8
)

data |>
  wald_test_nb(distribution = asymptotic())

#-------------------------------------------------------------------------
# simulated() examples
#-------------------------------------------------------------------------
data |>
  wald_test_nb(distribution = simulated(nsims = 200L))
```

---

eval_power_ci *Evaluate confidence intervals for power estimates*

Description

Calculates the confidence interval for a power estimate from a simulation study. The confidence interval quantifies uncertainty about the true power parameter.

When the number of simulations used to calculate a test's power is too small, the power estimate will have high uncertainty (wide confidence/prediction intervals). When the number of simulations used to calculate a test's power is too large, computational time may be prohibitive. This function allows

you to determine the appropriate number of simulated datasets to reach your desired precision for power before spending computational time on simulations.

Usage

```
eval_power_ci(power, nsims, ci_level = 0.95, method = c("wilson", "exact"))
```

Arguments

`power`	(numeric: `(0, 1)`) Hypothetical observed power value(s).
`nsims`	(integer: `[2, Inf)`) Number of simulations.
`ci_level`	(Scalar numeric: `0.95`; `(0,1)`) The confidence level.
`method`	(Scalar character: `"wilson"`; `c("wilson", "exact")`) Method for computing confidence intervals. One of `"wilson"` (default) or `"exact"`. See 'Details' for more information.

Details

Power estimation via simulation is a binomial proportion problem. The confidence interval answers: "What is the plausible range of true power values given my simulation results?"

Let π denote the hypothetical true power value, $\hat{\pi} = x/n$ denote the hypothetical observed power value, n denote the number of simulations, and $x = \text{round}(\hat{\pi} \cdot n)$ denote the number of rejections. Two methods are available.

Wilson Score Interval:

The Wilson score interval is derived from inverting the score test. Starting with the inequality

$$\left| \frac{\hat{\pi} - \pi}{\sqrt{\pi(1-\pi)/n}} \right| \leq z_{1-\alpha/2},$$

and solving the resulting quadratic for π yields

$$\frac{\hat{\pi} + \frac{z^2}{2n} \pm z\sqrt{\frac{\hat{\pi}(1-\hat{\pi})}{n} + \frac{z^2}{4n^2}}}{1 + \frac{z^2}{n}},$$

with $z = z_{1-\alpha/2}$ and $\hat{\pi} = x/n$.

Clopper-Pearson Interval:

The Clopper-Pearson exact interval inverts the binomial test via Beta quantiles. The bounds (π_L, π_U) satisfy:

$$P(X \geq x \mid \pi = \pi_L) = \alpha/2$$
$$P(X \leq x \mid \pi = \pi_U) = \alpha/2$$

With x successes in n trials,

$$\pi_L = B^{-1}\left(\frac{\alpha}{2}; x, n-x+1\right)$$
$$\pi_U = B^{-1}\left(1-\frac{\alpha}{2}; x+1, n-x\right)$$

where $B^{-1}(q; a, b)$ is the q-th quantile of Beta(a, b).

This method guarantees at least nominal coverage but is conservative (intervals are wider than necessary).

Approximate parametric tests:

When power is computed using approximate parametric tests (see `simulated()`), the power estimate and confidence/prediction intervals apply to the Monte Carlo test power $\mu_K = P(\hat{p} \leq \alpha)$ rather than the exact test power $\pi = P(p \leq \alpha)$. These quantities converge as the number of datasets simulated under the null hypothesis K increases. The minimum observable p-value is $1/(K+1)$, so $K > 1/\alpha - 1$ is required to observe any rejections. For practical accuracy, we recommend choosing $\max(5000, K \gg 1/\alpha - 1)$ for most scenarios. For example, if $\alpha = 0.05$, use `simulated(nsims = 5000)`.

Value

A list with elements:

Name	Description
`lower`	Lower bound of confidence interval.
`upper`	Upper bound of confidence interval.

References

Newcombe RG (1998). "Two-sided confidence intervals for the single proportion: comparison of seven methods." *Statistics in Medicine*, **17**(8), 857–872. ISSN 0277-6715, 1097-0258, doi:10.1002/(SICI)10970258(19980430)17:8<857::AIDSIM777>3.0.CO;2E.,

Wilson EB (1927). "Probable Inference, the Law of Succession, and Statistical Inference." *Journal of the American Statistical Association*, **22**(158), 209–212. ISSN 0162-1459, 1537-274X, doi:10.1080/01621459.1927.10502953.,

Clopper CJ, Pearson ES (1934). "THE USE OF CONFIDENCE OR FIDUCIAL LIMITS ILLUSTRATED IN THE CASE OF THE BINOMIAL." *Biometrika*, **26**(4), 404–413. ISSN 0006-3444, 1464-3510, doi:10.1093/biomet/26.4.404.

See Also

`add_power_ci()`, `eval_power_pi()`

Examples

```
#-------------------------------------------------------------------------
# eval_power_ci() examples
#-------------------------------------------------------------------------
library(depower)
```

```
# Expected CI for 80% power with 1000 simulations
eval_power_ci(power = 0.80, nsims = 1000)

# Compare precision across different simulation counts
eval_power_ci(power = 0.80, nsims = c(100, 500, 1000, 5000))

# Compare Wilson vs exact method
eval_power_ci(power = 0.80, nsims = 1000, method = "wilson")
eval_power_ci(power = 0.80, nsims = 1000, method = "exact")

# Vectorized over power values
eval_power_ci(power = c(0.70, 0.80, 0.90), nsims = 1000)

# 99% confidence interval
eval_power_ci(power = 0.80, nsims = 1000, ci_level = 0.99)
```

`eval_power_pi` *Evaluate Bayesian posterior predictive intervals for power estimates*

Description

Calculates the Bayesian posterior predictive interval for a power estimate from a simulation study. The posterior predictive interval quantifies the expected range of power estimates from a future simulation study.

When the number of simulations used to calculate a test's power is too small, the power estimate will have high uncertainty (wide confidence/prediction intervals). When the number of simulations used to calculate a test's power is too large, computational time may be prohibitive. This function allows you to determine the appropriate number of simulated datasets to reach your desired precision for power before spending computational time on simulations.

Usage

```
eval_power_pi(
  power,
  nsims,
  future_nsims = NULL,
  pi_level = 0.95,
  prior = c(1, 1)
)
```

Arguments

`power`	(numeric: `(0, 1)`) Hypothetical power value(s).
`nsims`	(integer: `[2, Inf)`) Number of simulations.

`future_nsims` (integer or `NULL`: `NULL`; `[2, Inf)`)
Number of simulations in the future study. If `NULL` (default), uses the same number as `nsims`.

`pi_level` (Scalar numeric: `0.95`; `(0,1)`)
The posterior predictive interval level.

`prior` (numeric vector of length 2: `c(1, 1)`; each `(0, Inf)`)
Parameters (α, β) for the Beta prior on true power. Default `c(1, 1)` is the uniform prior. Use `c(0.5, 0.5)` for the Jeffreys prior.

Details

Power estimation via simulation is a binomial proportion problem. The posterior predictive interval answers: "If I run a new simulation study with m simulations, what range of power estimates might I observe?"

Let π denote the hypothetical true power value, $\hat{\pi} = x/n$ denote the hypothetical observed power value, n denote the number of simulations, and $x = \text{round}(\hat{\pi} \cdot n)$ denote the number of rejections. With a Beta(α, β) prior on the true power π, the posterior after observing x successes in n trials is:

$$\pi \mid X = x \sim \text{Beta}(\alpha + x, \beta + n - x).$$

The posterior predictive distribution for Y, the number of successes in a future study with m trials, is Beta-Binomial:

$$Y \mid X = x \sim \text{BetaBinomial}(m, \alpha + x, \beta + n - x).$$

The posterior predictive interval is constructed from quantiles of this distribution, expressed as proportions Y/m.

The posterior predictive mean and variance of $\hat{\pi}_{\text{new}} = Y/m$ are:

$$E[\hat{\pi}_{\text{new}} \mid X = x] = \frac{\alpha + x}{\alpha + \beta + n}$$

$$\text{Var}[\hat{\pi}_{\text{new}} \mid X = x] = \frac{(\alpha + x)(\beta + n - x)(\alpha + \beta + n + m)}{m(\alpha + \beta + n)^2(\alpha + \beta + n + 1)}.$$

Argument `future_nsims`:

The argument `future_nsims` allows you to estimate prediction interval bounds for a hypothetical future study with different number of simulations. Note that a small initial number for `nsims` results in substantial uncertainty about the true power. A correspondingly large number of future simulations `future_nsims` will more precisely estimate the true power, but the past large uncertainty is still carried forward. Therefore you still need an adequate number of simulations `nsims` in the original study, not just more in the replication `future_nsims`, to ensure narrow prediction intervals.

Approximate parametric tests:

When power is computed using approximate parametric tests (see `simulated()`), the power estimate and confidence/prediction intervals apply to the Monte Carlo test power $\mu_K = P(\hat{p} \leq \alpha)$ rather than the exact test power $\pi = P(p \leq \alpha)$. These quantities converge as the number of

datasets simulated under the null hypothesis K increases. The minimum observable p-value is $1/(K+1)$, so $K > 1/\alpha - 1$ is required to observe any rejections. For practical accuracy, we recommend choosing $\max(5000, K \gg 1/\alpha - 1)$ for most scenarios. For example, if $\alpha = 0.05$, use `simulated(nsims = 5000)`.

Value

A list with elements:

Name	Description
`mean`	Predictive mean of future power estimate.
`lower`	Lower bound of posterior predictive interval.
`upper`	Upper bound of posterior predictive interval.

References

Gelman A, Carlin JB, Stern HS, Dunson DB, Vehtari A, Rubin DB (2013). *Bayesian data analysis*, Texts in statistical science series, Third edition edition. CRC Press, Taylor & Francis Group. ISBN 9781439840955.

See Also

`add_power_pi()`, `eval_power_ci()`

Examples

```
#------------------------------------------------------------------------
# eval_power_pi() examples
#------------------------------------------------------------------------
library(depower)

# Expected PI for 80% power with 1000 simulations
eval_power_pi(power = 0.80, nsims = 1000)

# Compare precision across different simulation counts
eval_power_pi(power = 0.80, nsims = c(100, 500, 1000, 5000))

# Predict for a larger future study (narrower interval)
eval_power_pi(power = 0.80, nsims = 1000, future_nsims = 5000)

# Predict for a smaller future study (wider interval)
eval_power_pi(power = 0.80, nsims = 1000, future_nsims = 100)

# Vectorized over power values
eval_power_pi(power = c(0.70, 0.80, 0.90), nsims = 1000)

# Use Jeffreys prior instead of uniform
eval_power_pi(power = 0.80, nsims = 1000, prior = c(0.5, 0.5))

# 99% predictive interval
eval_power_pi(power = 0.80, nsims = 1000, pi_level = 0.99)
```

glmm_bnb *GLMM for BNB ratio of means*

Description

Generalized linear mixed model for bivariate negative binomial outcomes.

Usage

```
glmm_bnb(data, test = "wald", ci_level = NULL, ...)
```

Arguments

data	(list) A list whose first element is the vector of negative binomial values from sample 1 and the second element is the vector of negative binomial values from sample 2. Each vector must be sorted by the subject/item index and must be the same sample size. NAs are silently excluded. The default output from `sim_bnb()`.
test	(String: `"wald"`; `c("wald", "lrt")`) The statistical method used for the test results. `test = "wald"` performs a Wald test and optionally the Wald confidence intervals. `test = "lrt"` performs a likelihood ratio test and optionally the profile likelihood confidence intervals (means and ratio). The Wald confidence interval is always used for the limits of the mean of sample 2, dispersion, and standard deviation of the item (subject) random intercept.
ci_level	(Scalar numeric: `NULL`; `(0, 1)`) If `NULL`, confidence intervals are set as `NA`. If in `(0, 1)`, confidence intervals are calculated at the specified level. Profile likelihood intervals are computationally intensive, so intervals from `test = "lrt"` may be slow.
...	Optional arguments passed to `glmmTMB::glmmTMB()`.

Details

Uses `glmmTMB::glmmTMB()` in the form

```
glmmTMB(
  formula = value ~ condition + (1 | item),
  data = data,
  dispformula = ~ 1,
  family = nbinom2
)
```

to model dependent negative binomial outcomes $X_1, X_2 \sim \text{BNB}(\mu, r, \theta)$ where μ is the mean of sample 1, r is the ratio of the means of sample 2 with respect to sample 1, and θ is the dispersion parameter.

The hypotheses for the LRT and Wald test of r are

$$H_{null} : log(r) = 0$$
$$H_{alt} : log(r) \neq 0$$

where $r = \frac{\bar{X}_2}{\bar{X}_1}$ is the population ratio of arithmetic means for sample 2 with respect to sample 1 and $log(r_{null}) = 0$ assumes the population means are identical.

When simulating data from `sim_bnb()`, the mean is a function of the item (subject) random effect which in turn is a function of the dispersion parameter. Thus, `glmm_bnb()` has biased mean and dispersion estimates. The bias increases as the dispersion parameter gets smaller and decreases as the dispersion parameter gets larger. However, estimates of the ratio and standard deviation of the random intercept tend to be accurate. The p-value for `glmm_bnb()` is generally overconservative compared to `glmm_poisson()`, `wald_test_bnb()` and `lrt_bnb()`. In summary, the negative binomial mixed-effects model fit by `glmm_bnb()` is not recommended for the BNB data simulated by `sim_bnb()`. Instead, `wald_test_bnb()` or `lrt_bnb()` should typically be used instead.

Value

A list with the following elements:

Slot	Subslot	Name	Description
1		`chisq`	χ^2 test statistic for the ratio of means.
2		`df`	Degrees of freedom.
3		`p`	p-value.
4		`ratio`	Estimated ratio of means (sample 2 / sample 1).
4	1	`estimate`	Point estimate.
4	2	`lower`	Confidence interval lower bound.
4	3	`upper`	Confidence interval upper bound.
5		`mean1`	Estimated mean of sample 1.
5	1	`estimate`	Point estimate.
5	2	`lower`	Confidence interval lower bound.
5	3	`upper`	Confidence interval upper bound.
6		`mean2`	Estimated mean of sample 2.
6	1	`estimate`	Point estimate.
6	2	`lower`	Wald confidence interval lower bound.
6	3	`upper`	Wald confidence interval upper bound.
7		`dispersion`	Estimated dispersion.
7	1	`estimate`	Point estimate.
7	2	`lower`	Confidence interval lower bound.
7	3	`upper`	Confidence interval upper bound.
8		`item_sd`	Estimated standard deviation of the item (subject) random intercept.
8	1	`estimate`	Point estimate.
8	2	`lower`	Confidence interval lower bound.
8	3	`upper`	Confidence interval upper bound.
9		`n1`	Sample size of sample 1.
10		`n2`	Sample size of sample 2.
11		`method`	Method used for the results.
12		`test`	Type of hypothesis test.
13		`alternative`	The alternative hypothesis.

14	`ci_level`	Confidence level of the interval.
15	`hessian`	Information about the Hessian matrix.
16	`convergence`	Information about convergence.

References

Hilbe JM (2011). *Negative Binomial Regression*, 2 edition. Cambridge University Press. ISBN 9780521198158 9780511973420, doi:10.1017/CBO9780511973420.

Hilbe JM (2014). *Modeling count data*. Cambridge University Press, New York, NY. ISBN 9781107028333 9781107611252, doi:10.1017/CBO9781139236065.

See Also

`wald_test_bnb()`, `lrt_bnb()`, `glmm_poisson()`

Examples

```
#------------------------------------------------------------------------------
# glmm_bnb() examples
#------------------------------------------------------------------------------
library(depower)

set.seed(1234)
d <- sim_bnb(
  n = 40,
  mean1 = 10,
  ratio = 1.2,
  dispersion = 2
)

lrt <- glmm_bnb(d, test = "lrt")
lrt

wald <- glmm_bnb(d, test = "wald", ci_level = 0.95)
wald

#------------------------------------------------------------------------------
# Compare results to manual calculation of chi-square statistic
#------------------------------------------------------------------------------
# Use the same data, but as a data frame instead of list
set.seed(1234)
d <- sim_bnb(
  n = 40,
  mean1 = 10,
  ratio = 1.2,
  dispersion = 2,
  return_type = "data.frame"
)

mod_alt <- glmmTMB::glmmTMB(
  formula = value ~ condition + (1 | item),
  data = d,
```

```
  dispformula = ~ 1,
  family = glmmTMB::nbinom2
)
mod_null <- glmmTMB::glmmTMB(
  formula = value ~ 1 + (1 | item),
  data = d,
  dispformula = ~ 1,
  family = glmmTMB::nbinom2
)

lrt_chisq <- as.numeric(-2 * (logLik(mod_null) - logLik(mod_alt)))
lrt_chisq
wald_chisq <- summary(mod_alt)$coefficients$cond["condition2", "z value"]^2
wald_chisq

anova(mod_null, mod_alt)
```

`glmm_poisson` *GLMM for Poisson ratio of means*

Description

Generalized linear mixed model for two dependent Poisson outcomes.

Usage

```
glmm_poisson(data, test = "wald", ci_level = NULL, ...)
```

Arguments

`data`	(list) A list whose first element is the vector of Poisson values from sample 1 and the second element is the vector of Poisson values from sample 2. Each vector must be sorted by the subject/item index and must be the same sample size. NAs are silently excluded. The default output from `sim_bnb()`.
`test`	(String: `"wald"`; `c("wald", "lrt")`) The statistical method used for the test results. `test = "wald"` performs a Wald test and optionally the Wald confidence intervals. `test = "lrt"` performs a likelihood ratio test and optionally the profile likelihood confidence intervals (means and ratio). The Wald confidence interval is always used for the limits of the mean of sample 2 and standard deviation of the item (subject) random intercept.
`ci_level`	(Scalar numeric: `NULL`; `(0, 1)`) If `NULL`, confidence intervals are set as `NA`. If in `(0, 1)`, confidence intervals are calculated at the specified level. Profile likelihood intervals are computationally intensive, so intervals from `test = "lrt"` may be slow.
`...`	Optional arguments passed to `glmmTMB::glmmTMB()`.

Details

Uses `glmmTMB::glmmTMB()` in the form

```
glmmTMB(
  formula = value ~ condition + (1 | item),
  data = data,
  family = stats::poisson
)
```

to model dependent Poisson outcomes $X_1 \sim \text{Poisson}(\mu)$ and $X_2 \sim \text{Poisson}(r\mu)$ where μ is the mean of sample 1 and r is the ratio of the means of sample 2 with respect to sample 1.

The hypotheses for the LRT and Wald test of r are

$$H_{null} : log(r) = 0$$
$$H_{alt} : log(r) \neq 0$$

where $r = \frac{\bar{X}_2}{\bar{X}_1}$ is the population ratio of arithmetic means for sample 2 with respect to sample 1 and $log(r_{null}) = 0$ assumes the population means are identical.

When simulating data from `sim_bnb()`, the mean is a function of the item (subject) random effect which in turn is a function of the dispersion parameter. Thus, `glmm_poisson()` has biased mean estimates. The bias increases as the dispersion parameter gets smaller and decreases as the dispersion parameter gets larger. However, estimates of the ratio and standard deviation of the random intercept tend to be accurate. In summary, the Poisson mixed-effects model fit by `glmm_poisson()` is not recommended for the BNB data simulated by `sim_bnb()`. Instead, `wald_test_bnb()` or `lrt_bnb()` should typically be used instead.

Value

A list with the following elements:

Slot	Subslot	Name	Description
1		`chisq`	χ^2 test statistic for the ratio of means.
2		`df`	Degrees of freedom.
3		`p`	p-value.
4		`ratio`	Estimated ratio of means (sample 2 / sample 1).
4	1	`estimate`	Point estimate.
4	2	`lower`	Confidence interval lower bound.
4	3	`upper`	Confidence interval upper bound.
5		`mean1`	Estimated mean of sample 1.
5	1	`estimate`	Point estimate.
5	2	`lower`	Confidence interval lower bound.
5	3	`upper`	Confidence interval upper bound.
6		`mean2`	Estimated mean of sample 2.
6	1	`estimate`	Point estimate.
6	2	`lower`	Wald confidence interval lower bound.
6	3	`upper`	Wald confidence interval upper bound.
7		`item_sd`	Estimated standard deviation of the item (subject) random intercept.

7	1	estimate	Point estimate.
7	2	lower	Confidence interval lower bound.
7	3	upper	Confidence interval upper bound.
8		n1	Sample size of sample 1.
9		n2	Sample size of sample 2.
10		method	Method used for the results.
11		test	Type of hypothesis test.
12		alternative	The alternative hypothesis.
13		ci_level	Confidence level of the interval.
14		hessian	Information about the Hessian matrix.
15		convergence	Information about convergence.

References

Hilbe JM (2011). *Negative Binomial Regression*, 2 edition. Cambridge University Press. ISBN 9780521198158 9780511973420, doi:10.1017/CBO9780511973420.

Hilbe JM (2014). *Modeling count data*. Cambridge University Press, New York, NY. ISBN 9781107028333 9781107611252, doi:10.1017/CBO9781139236065.

See Also

wald_test_bnb(), lrt_bnb(), glmm_bnb()

Examples

```
#------------------------------------------------------------------------------
# glmm_poisson() examples
#------------------------------------------------------------------------------
library(depower)

set.seed(1234)
d <- sim_bnb(
  n = 40,
  mean1 = 10,
  ratio = 1.2,
  dispersion = 2
)

lrt <- glmm_poisson(d, test = "lrt")
lrt

wald <- glmm_poisson(d, test = "wald", ci_level = 0.95)
wald

#------------------------------------------------------------------------------
# Compare results to manual calculation of chi-square statistic
#------------------------------------------------------------------------------
# Use the same data, but as a data frame instead of list
set.seed(1234)
d <- sim_bnb(
  n = 40,
```

```
  mean1 = 10,
  ratio = 1.2,
  dispersion = 2,
  return_type = "data.frame"
)

mod_alt <- glmmTMB::glmmTMB(
  formula = value ~ condition + (1 | item),
  data = d,
  family = stats::poisson
)
mod_null <- glmmTMB::glmmTMB(
  formula = value ~ 1 + (1 | item),
  data = d,
  family = stats::poisson
)

lrt_chisq <- as.numeric(-2 * (logLik(mod_null) - logLik(mod_alt)))
lrt_chisq
wald_chisq <- summary(mod_alt)$coefficients$cond["condition2", "z value"]^2
wald_chisq

anova(mod_null, mod_alt)
```

`glm_nb` *GLM for NB ratio of means*

Description

Generalized linear model for two independent negative binomial outcomes.

Usage

```
glm_nb(data, equal_dispersion = FALSE, test = "wald", ci_level = NULL, ...)
```

Arguments

`data`	(list) A list whose first element is the vector of negative binomial values from group 1 and the second element is the vector of negative binomial values from group 2. NAs are silently excluded. The default output from `sim_nb()`.
`equal_dispersion`	(Scalar logical: `FALSE`) If `TRUE`, the model is fit assuming both groups have the same population dispersion parameter. If `FALSE` (default), the model is fit assuming different dispersions.

`test` (String: `"wald"`; `c("wald", "lrt")`)
The statistical method used for the test results. `test = "wald"` performs a Wald test and optionally the Wald confidence intervals. `test = "lrt"` performs a likelihood ratio test and optionally the profile likelihood confidence intervals. Wald confidence intervals are always returned for the limits of the mean of sample 2 and, when `equal_dispersion=FALSE`, for the limits of the dispersion of sample 2.

`ci_level` (Scalar numeric: `NULL`; `(0, 1)`)
If `NULL`, confidence intervals are set as `NA`. If in `(0, 1)`, confidence intervals are calculated at the specified level. Profile likelihood intervals are computationally intensive, so intervals from `test = "lrt"` may be slow.

`...` Optional arguments passed to `glmmTMB::glmmTMB()`.

Details

Uses `glmmTMB::glmmTMB()` in the form

```
glmmTMB(
  formula = value ~ condition,
  data = data,
  dispformula = ~ condition,
  family = nbinom2
)
```

to model independent negative binomial outcomes $X_1 \sim \text{NB}(\mu, \theta_1)$ and $X_2 \sim \text{NB}(r\mu, \theta_2)$ where μ is the mean of group 1, r is the ratio of the means of group 2 with respect to group 1, θ_1 is the dispersion parameter of group 1, and θ_2 is the dispersion parameter of group 2.

The hypotheses for the LRT and Wald test of r are

$$H_{null} : log(r) = 0$$
$$H_{alt} : log(r) \neq 0$$

where $r = \frac{\bar{X}_2}{\bar{X}_1}$ is the population ratio of arithmetic means for group 2 with respect to group 1 and $log(r_{null}) = 0$ assumes the population means are identical.

Value

A list with the following elements:

Slot	Subslot	Name	Description
1		`chisq`	χ^2 test statistic for the ratio of means.
2		`df`	Degrees of freedom.
3		`p`	p-value.
4		`ratio`	Estimated ratio of means (group 2 / group 1).
4	1	`estimate`	Point estimate.
4	2	`lower`	Confidence interval lower bound.
4	3	`upper`	Confidence interval upper bound.

5		mean1	Estimated mean of group 1.
5	1	estimate	Point estimate.
5	2	lower	Confidence interval lower bound.
5	3	upper	Confidence interval upper bound.
6		mean2	Estimated mean of group 2.
6	1	estimate	Point estimate.
6	2	lower	Wald confidence interval lower bound.
6	3	upper	Wald confidence interval upper bound.
7		dispersion1	Estimated dispersion of group 1.
7	1	estimate	Point estimate.
7	2	lower	Confidence interval lower bound.
7	3	upper	Confidence interval upper bound.
8		dispersion2	Estimated dispersion of group 2.
8	1	estimate	Point estimate.
8	2	lower	Wald confidence interval lower bound.
8	3	upper	Wald confidence interval upper bound.
9		n1	Sample size of group 1.
10		n2	Sample size of group 2.
11		method	Method used for the results.
12		test	Type of hypothesis test.
13		alternative	The alternative hypothesis.
14		equal_dispersion	Whether or not equal dispersions were assumed.
15		ci_level	Confidence level of the intervals.
16		hessian	Information about the Hessian matrix.
17		convergence	Information about convergence.

References

Hilbe JM (2011). *Negative Binomial Regression*, 2 edition. Cambridge University Press. ISBN 9780521198158 9780511973420, doi:10.1017/CBO9780511973420.

Hilbe JM (2014). *Modeling count data*. Cambridge University Press, New York, NY. ISBN 9781107028333 9781107611252, doi:10.1017/CBO9781139236065.

See Also

wald_test_nb(), lrt_nb()

Examples

```
#------------------------------------------------------------------------
# glm_nb() examples
#------------------------------------------------------------------------
library(depower)

set.seed(1234)
d <- sim_nb(
  n1 = 60,
  n2 = 40,
  mean1 = 10,
  ratio = 1.5,
```

```
  dispersion1 = 2,
  dispersion2 = 8
)

lrt <- glm_nb(d, equal_dispersion = FALSE, test = "lrt", ci_level = 0.95)
lrt

wald <- glm_nb(d, equal_dispersion = FALSE, test = "wald", ci_level = 0.95)
wald

#------------------------------------------------------------------------------
# Compare results to manual calculation of chi-square statistic
#------------------------------------------------------------------------------
# Use the same data, but as a data frame instead of list
set.seed(1234)
df <- sim_nb(
  n1 = 60,
  n2 = 40,
  mean1 = 10,
  ratio = 1.5,
  dispersion1 = 2,
  dispersion2 = 8,
  return_type = "data.frame"
)

mod_alt <- glmmTMB::glmmTMB(
  formula = value ~ condition,
  data = df,
  dispformula = ~ condition,
  family = glmmTMB::nbinom2
)
mod_null <- glmmTMB::glmmTMB(
  formula = value ~ 1,
  data = df,
  dispformula = ~ condition,
  family = glmmTMB::nbinom2
)

lrt_chisq <- as.numeric(-2 * (logLik(mod_null) - logLik(mod_alt)))
lrt_chisq
wald_chisq <- summary(mod_alt)$coefficients$cond["condition2", "z value"]^2
wald_chisq

anova(mod_null, mod_alt)

#------------------------------------------------------------------------------
# Compare results to wald_test_nb()
#------------------------------------------------------------------------------
wald2 <- wald_test_nb(d, equal_dispersion = FALSE, ci_level = 0.95)
all.equal(wald$chisq, wald2$chisq, tolerance = 0.01)
```

`lrt_bnb` *Likelihood ratio test for BNB ratio of means*

Description

Likelihood ratio test for the ratio of means from bivariate negative binomial outcomes.

Usage

```
lrt_bnb(data, ratio_null = 1, distribution = asymptotic(), ...)
```

Arguments

`data`	(list) A list whose first element is the vector of negative binomial values from sample 1 and the second element is the vector of negative binomial values from sample 2. Each vector must be sorted by the subject/item index and must be the same sample size. NAs are silently excluded. The default output from `sim_bnb()`.
`ratio_null`	(Scalar numeric: `1`; `(0, Inf)`) The ratio of means assumed under the null hypothesis (sample 2 / sample 1). Typically, `ratio_null = 1` (no difference). See 'Details' for additional information.
`distribution`	(function: `asymptotic()` or `simulated()`) The method used to define the distribution of the χ^2 likelihood ratio test statistic under the null hypothesis. See 'Details' and `asymptotic()` or `simulated()` for additional information.
`...`	Optional arguments passed to the MLE function `mle_bnb()`.

Details

This function is primarily designed for speed in simulation. Missing values are silently excluded.

Suppose $X_1 \mid G = g \sim \text{Poisson}(\mu g)$ and $X_2 \mid G = g \sim \text{Poisson}(r\mu g)$ where $G \sim \text{Gamma}(\theta, \theta^{-1})$ is the random item (subject) effect. Then $X_1, X_2 \sim \text{BNB}(\mu, r, \theta)$ is the joint distribution where X_1 and X_2 are dependent (though conditionally independent), X_1 is the count outcome for sample 1 of the items (subjects), X_2 is the count outcome for sample 2 of the items (subjects), μ is the conditional mean of sample 1, r is the ratio of the conditional means of sample 2 with respect to sample 1, and θ is the gamma distribution shape parameter which controls the dispersion and the correlation between sample 1 and 2.

The hypotheses for the LRT of r are

$$H_{null} : r = r_{null}$$
$$H_{alt} : r \neq r_{null}$$

where $r = \frac{\bar{X}_2}{\bar{X}_1}$ is the population ratio of arithmetic means for sample 2 with respect to sample 1 and r_{null} is a constant for the assumed null population ratio of means (typically $r_{null} = 1$).

The LRT statistic is

$$\begin{aligned}\lambda &= -2\ln\frac{\sup_{\Theta_{null}} L(r,\mu,\theta)}{\sup_{\Theta} L(r,\mu,\theta)} \\ &= -2\left[\ln\sup_{\Theta_{null}} L(r,\mu,\theta) - \ln\sup_{\Theta} L(r,\mu,\theta)\right] \\ &= -2(l(r_{null},\tilde{\mu},\tilde{\theta}) - l(\hat{r},\hat{\mu},\hat{\theta}))\end{aligned}$$

Under H_{null}, the LRT test statistic is asymptotically distributed as χ^2_1. The approximate level α test rejects H_{null} if $\lambda \geq \chi^2_1(1-\alpha)$. However, the asymptotic critical value is known to underestimate the exact critical value and the nominal significance level may not be achieved for small sample sizes. Argument `distribution` allows control of the distribution of the χ^2_1 test statistic under the null hypothesis by use of functions `asymptotic()` and `simulated()`.

Value

A list with the following elements:

Slot	Subslot	Name	Description
1		`chisq`	χ^2 test statistic for the ratio of means.
2		`df`	Degrees of freedom.
3		`p`	p-value.
4		`ratio`	Estimated ratio of means (sample 2 / sample 1).
5		`alternative`	Point estimates under the alternative hypothesis.
5	1	`mean1`	Estimated mean of sample 1.
5	2	`mean2`	Estimated mean of sample 2.
5	3	`dispersion`	Estimated dispersion.
6		`null`	Point estimates under the null hypothesis.
6	1	`mean1`	Estimated mean of sample 1.
6	2	`mean2`	Estimated mean of sample 2.
6	3	`dispersion`	Estimated dispersion.
7		`n1`	The sample size of sample 1.
8		`n2`	The sample size of sample 2.
9		`method`	Method used for the results.
10		`ratio_null`	Assumed population ratio of means.
11		`mle_code`	Integer indicating why the optimization process terminated.
12		`mle_message`	Information from the optimizer.

References

Rettiganti M, Nagaraja HN (2012). "Power Analyses for Negative Binomial Models with Application to Multiple Sclerosis Clinical Trials." *Journal of Biopharmaceutical Statistics*, **22**(2), 237–259. ISSN 1054-3406, 1520-5711, doi:10.1080/10543406.2010.528105.

Aban IB, Cutter GR, Mavinga N (2009). "Inferences and power analysis concerning two negative binomial distributions with an application to MRI lesion counts data." *Computational Statistics & Data Analysis*, **53**(3), 820–833. ISSN 01679473, doi:10.1016/j.csda.2008.07.034.

See Also

`wald_test_bnb()`

Examples

```
#-------------------------------------------------------------------------
# lrt_bnb() examples
#-------------------------------------------------------------------------
library(depower)

set.seed(1234)
sim_bnb(
  n = 40,
  mean1 = 10,
  ratio = 1.2,
  dispersion = 2
) |>
  lrt_bnb()
```

`lrt_nb` *Likelihood ratio test for NB ratio of means*

Description

Likelihood ratio test for the ratio of means from two independent negative binomial outcomes.

Usage

```
lrt_nb(
  data,
  equal_dispersion = FALSE,
  ratio_null = 1,
  distribution = asymptotic(),
  ...
)
```

Arguments

`data`	(list) A list whose first element is the vector of negative binomial values from group 1 and the second element is the vector of negative binomial values from group 2. NAs are silently excluded. The default output from `sim_nb()`.
`equal_dispersion`	(Scalar logical: `FALSE`) If `TRUE`, the LRT is calculated assuming both groups have the same population dispersion parameter. If `FALSE` (default), the LRT is calculated assuming different dispersions.
`ratio_null`	(Scalar numeric: `1`; `(0, Inf)`) The ratio of means assumed under the null hypothesis (group 2 / group 1). Typically `ratio_null = 1` (no difference). See 'Details' for additional information.

`distribution` (function: `asymptotic()` or `simulated()`)
The method used to define the distribution of the χ^2 likelihood ratio test statistic under the null hypothesis. See 'Details' and `asymptotic()` or `simulated()` for additional information.

`...` Optional arguments passed to the MLE function `mle_nb()`.

Details

This function is primarily designed for speed in simulation. Missing values are silently excluded.

Suppose $X_1 \sim NB(\mu, \theta_1)$ and $X_2 \sim NB(r\mu, \theta_2)$ where X_1 and X_2 are independent, X_1 is the count outcome for items in group 1, X_2 is the count outcome for items in group 2, μ is the arithmetic mean count in group 1, r is the ratio of arithmetic means for group 2 with respect to group 1, θ_1 is the dispersion parameter of group 1, and θ_2 is the dispersion parameter of group 2.

The hypotheses for the LRT of r are

$$H_{null} : r = r_{null}$$
$$H_{alt} : r \neq r_{null}$$

where $r = \frac{\bar{X}_2}{\bar{X}_1}$ is the population ratio of arithmetic means for group 2 with respect to group 1 and r_{null} is a constant for the assumed null population ratio of means (typically $r_{null} = 1$).

The LRT statistic is

$$\begin{aligned}\lambda &= -2 \ln \frac{\sup_{\Theta_{null}} L(r, \mu, \theta_1, \theta_2)}{\sup_{\Theta} L(r, \mu, \theta_1, \theta_2)} \\ &= -2 \left[\ln \sup_{\Theta_{null}} L(r, \mu, \theta_1, \theta_2) - \ln \sup_{\Theta} L(r, \mu, \theta_1, \theta_2)\right] \\ &= -2(l(r_{null}, \tilde{\mu}, \tilde{\theta}_1, \tilde{\theta}_2) - l(\hat{r}, \hat{\mu}, \hat{\theta}_1, \hat{\theta}_2))\end{aligned}$$

Under H_{null}, the LRT test statistic is asymptotically distributed as χ^2_1. The approximate level α test rejects H_{null} if $\lambda \geq \chi^2_1(1-\alpha)$. However, the asymptotic critical value is known to underestimate the exact critical value and the nominal significance level may not be achieved for small sample sizes. Argument `distribution` allows control of the distribution of the χ^2_1 test statistic under the null hypothesis by use of functions `asymptotic()` and `simulated()`.

Note that standalone use of this function with `equal_dispersion = FALSE` and `distribution = simulated()`, e.g.

```
data |>
  lrt_nb(
    equal_dispersion = FALSE,
    distribution = simulated()
  )
```

results in a nonparametric randomization test based on label permutation. This violates the assumption of exchangeability for the randomization test because the labels are not exchangeable when the null hypothesis assumes unequal dispersions. However, used inside `power()`, e.g.

```
data |>
  power(
    lrt_nb(
      equal_dispersion = FALSE,
      distribution = simulated()
    )
  )
```

results in parametric resampling and no label permutation is performed. Thus, setting `equal_dispersion = FALSE` and `distribution = simulated()` is only recommended when `lrt_nb()` is used inside of `power()`. See also, `simulated()`.

Value

A list with the following elements:

Slot	Subslot	Name	Description
1		`chisq`	χ^2 test statistic for the ratio of means.
2		`df`	Degrees of freedom.
3		`p`	p-value.
4		`ratio`	Estimated ratio of means (group 2 / group 1).
5		`alternative`	Point estimates under the alternative hypothesis.
5	1	`mean1`	Estimated mean of group 1.
5	2	`mean2`	Estimated mean of group 2.
5	3	`dispersion1`	Estimated dispersion of group 1.
5	4	`dispersion2`	Estimated dispersion of group 2.
6		`null`	Point estimates under the null hypothesis.
6	1	`mean1`	Estimated mean of group 1.
6	2	`mean2`	Estimated mean of group 2.
6	3	`dispersion1`	Estimated dispersion of group 1.
6	4	`dispersion2`	Estimated dispersion of group 2.
7		`n1`	Sample size of group 1.
8		`n2`	Sample size of group 2.
9		`method`	Method used for the results.
10		`equal_dispersion`	Whether or not equal dispersions were assumed.
11		`ratio_null`	Assumed population ratio of means.
12		`mle_code`	Integer indicating why the optimization process terminated.
13		`mle_message`	Information from the optimizer.

References

Rettiganti M, Nagaraja HN (2012). "Power Analyses for Negative Binomial Models with Application to Multiple Sclerosis Clinical Trials." *Journal of Biopharmaceutical Statistics*, **22**(2), 237–259. ISSN 1054-3406, 1520-5711, doi:10.1080/10543406.2010.528105.

Aban IB, Cutter GR, Mavinga N (2009). "Inferences and power analysis concerning two negative binomial distributions with an application to MRI lesion counts data." *Computational Statistics & Data Analysis*, **53**(3), 820–833. ISSN 01679473, doi:10.1016/j.csda.2008.07.034.

See Also

wald_test_nb()

Examples

```
#-------------------------------------------------------------------------
# lrt_nb() examples
#-------------------------------------------------------------------------
library(depower)

set.seed(1234)
sim_nb(
  n1 = 60,
  n2 = 40,
  mean1 = 10,
  ratio = 1.5,
  dispersion1 = 2,
  dispersion2 = 8
) |>
  lrt_nb()
```

---

mle_bnb *MLE for BNB*

Description

Maximum likelihood estimates (MLE) for bivariate negative binomial outcomes.

Usage

```
mle_bnb_null(data, ratio_null = 1, method = "nlm_constrained", ...)

mle_bnb_alt(data, method = "nlm_constrained", ...)
```

Arguments

data	(list) A list whose first element is the vector of negative binomial values from sample 1 and the second element is the vector of negative binomial values from sample 2. Each vector must be sorted by the subject/item index and must be the same sample size. NAs are silently excluded. The default output from sim_bnb().
ratio_null	(Scalar numeric: 1; (0, Inf)) The ratio of means assumed under the null hypothesis (sample 2 / sample 1). Typically ratio_null = 1 (no difference).
method	(string: "nlm_constrained") The optimization method. Must choose one of "nlm", "nlm_constrained", "optim", or "optim_constrained". The default bounds for constrained optimization are [1e-03, 1e06].

... Optional arguments passed to the optimization method.

Details

These functions are primarily designed for speed in simulation. Missing values are silently excluded.

Suppose $X_1 \mid G = g \sim \text{Poisson}(\mu g)$ and $X_2 \mid G = g \sim \text{Poisson}(r\mu g)$ where $G \sim \text{Gamma}(\theta, \theta^{-1})$ is the random item (subject) effect. Then $X_1, X_2 \sim \text{BNB}(\mu, r, \theta)$ is the joint distribution where X_1 and X_2 are dependent (though conditionally independent), X_1 is the count outcome for sample 1 of the items (subjects), X_2 is the count outcome for sample 2 of the items (subjects), μ is the conditional mean of sample 1, r is the ratio of the conditional means of sample 2 with respect to sample 1, and θ is the gamma distribution shape parameter which controls the dispersion and the correlation between sample 1 and 2.

The MLEs of r and μ are $\hat{r} = \frac{\bar{x}_2}{\bar{x}_1}$ and $\hat{\mu} = \bar{x}_1$. The MLE of θ is found by maximizing the profile log-likelihood $l(\hat{r}, \hat{\mu}, \theta)$ with respect to θ. When $r = r_{null}$ is known, the MLE of μ is $\tilde{\mu} = \frac{\bar{x}_1 + \bar{x}_2}{1 + r_{null}}$ and $\tilde{\theta}$ is obtained by maximizing the profile log-likelihood $l(r_{null}, \tilde{\mu}, \theta)$ with respect to θ.

The backend method for numerical optimization is controlled by argument `method` which refers to `stats::nlm()`, `stats::nlminb()`, or `stats::optim()`. If you would like to see warnings from the optimizer, include argument `warnings = TRUE`.

Value

- For `mle_bnb_alt`, a list with the following elements:

Slot	Name	Description
1	`mean1`	MLE for mean of sample 1.
2	`mean2`	MLE for mean of sample 2.
3	`ratio`	MLE for ratio of means.
4	`dispersion`	MLE for BNB dispersion.
5	`nll`	Minimum of negative log-likelihood.
6	`nparams`	Number of estimated parameters.
7	`n1`	Sample size of sample 1.
8	`n2`	Sample size of sample 2.
9	`method`	Method used for the results.
10	`mle_method`	Method used for optimization.
11	`mle_code`	Integer indicating why the optimization process terminated.
12	`mle_message`	Additional information from the optimizer.

- For `mle_bnb_null`, a list with the following elements:

Slot	Name	Description
1	`mean1`	MLE for mean of sample 1.
2	`mean2`	MLE for mean of sample 2.
3	`ratio_null`	Population ratio of means assumed for null hypothesis. `mean2 = mean1 * ratio_null`.
4	`dispersion`	MLE for BNB dispersion.
5	`nll`	Minimum of negative log-likelihood.
6	`nparams`	Number of estimated parameters.

7	n1	Sample size of sample 1.
8	n2	Sample size of sample 2.
9	method	Method used for the results.
10	mle_method	Method used for optimization.
11	mle_code	Integer indicating why the optimization process terminated.
12	mle_message	Additional information from the optimizer.

References

Rettiganti M, Nagaraja HN (2012). "Power Analyses for Negative Binomial Models with Application to Multiple Sclerosis Clinical Trials." *Journal of Biopharmaceutical Statistics*, **22**(2), 237–259. ISSN 1054-3406, 1520-5711, doi:10.1080/10543406.2010.528105.

Aban IB, Cutter GR, Mavinga N (2009). "Inferences and power analysis concerning two negative binomial distributions with an application to MRI lesion counts data." *Computational Statistics & Data Analysis*, **53**(3), 820–833. ISSN 01679473, doi:10.1016/j.csda.2008.07.034.

See Also

sim_bnb(), nll_bnb

Examples

```
#-------------------------------------------------------------------------
# mle_bnb() examples
#-------------------------------------------------------------------------
library(depower)

set.seed(1234)
d <- sim_bnb(
  n = 40,
  mean1 = 10,
  ratio = 1.2,
  dispersion = 2
)

mle_alt <- d |>
  mle_bnb_alt()

mle_null <- d |>
  mle_bnb_null()

mle_alt
mle_null
```

`mle_nb` *MLE for NB*

Description

Maximum likelihood estimates (MLE) for two independent negative binomial outcomes.

Usage

```
mle_nb_null(
  data,
  equal_dispersion = FALSE,
  ratio_null = 1,
  method = "nlm_constrained",
  ...
)

mle_nb_alt(data, equal_dispersion = FALSE, method = "nlm_constrained", ...)
```

Arguments

`data`	(list) A list whose first element is the vector of negative binomial values from group 1 and the second element is the vector of negative binomial values from group 2. NAs are silently excluded. The default output from `sim_nb()`.
`equal_dispersion`	(Scalar logical: `FALSE`) If `TRUE`, the MLEs are calculated assuming both groups have the same population dispersion parameter. If `FALSE` (default), the MLEs are calculated assuming different dispersions.
`ratio_null`	(Scalar numeric: `1`; `(0, Inf)`) The ratio of means assumed under the null hypothesis (group 2 / group 1). Typically `ratio_null = 1` (no difference).
`method`	(string: `"nlm_constrained"`) The optimization method. Must choose one of `"nlm"`, `"nlm_constrained"`, `"optim"`, or `"optim_constrained"`. The default bounds for constrained optimization are `[1e-03, 1e06]`.
`...`	Optional arguments passed to the optimization method.

Details

These functions are primarily designed for speed in simulation. Missing values are silently excluded.

Suppose $X_1 \sim \text{NB}(\mu, \theta_1)$ and $X_2 \sim \text{NB}(r\mu, \theta_2)$, where X_1 and X_2 are independent, X_1 is the count outcome for items in group 1, X_2 is the count outcome for items in group 2, μ is the arithmetic

mean count in group 1, r is the ratio of arithmetic means for group 2 with respect to group 1, θ_1 is the dispersion parameter of group 1, and θ_2 is the dispersion parameter of group 2.

The MLEs of r and μ are $\hat{r} = \frac{\bar{x}_2}{\bar{x}_1}$ and $\hat{\mu} = \bar{x}_1$. The MLEs of θ_1 and θ_2 are found by maximizing the profile log-likelihood $l(\hat{r}, \hat{\mu}, \theta_1, \theta_2)$ with respect to θ_1 and θ_2. When $r = r_{null}$ is known, the MLE of μ is $\tilde{\mu} = \frac{n_1\bar{x}_1 + n_2\bar{x}_2}{n_1+n_2}$ and $\tilde{\theta}_1$ and $\tilde{\theta}_2$ are obtained by maximizing the profile log-likelihood $l(r_{null}, \tilde{\mu}, \theta_1, \theta_2)$.

The backend method for numerical optimization is controlled by argument `method` which refers to `stats::nlm()`, `stats::nlminb()`, or `stats::optim()`. If you would like to see warnings from the optimizer, include argument `warnings = TRUE`.

Value

- For `mle_nb_alt()`, a list with the following elements:

Slot	Name	Description
1	`mean1`	MLE for mean of group 1.
2	`mean2`	MLE for mean of group 2.
3	`ratio`	MLE for ratio of means.
4	`dispersion1`	MLE for dispersion of group 1.
5	`dispersion2`	MLE for dispersion of group 2.
6	`equal_dispersion`	Were equal dispersions assumed.
7	`n1`	Sample size of group 1.
8	`n2`	Sample size of group 2.
9	`nll`	Minimum of negative log-likelihood.
10	`nparams`	Number of estimated parameters.
11	`method`	Method used for the results.
12	`mle_method`	Method used for optimization.
13	`mle_code`	Integer indicating why the optimization process terminated.
14	`mle_message`	Additional information from the optimizer.

- For `mle_nb_null()`, a list with the following elements:

Slot	Name	Description
1	`mean1`	MLE for mean of group 1.
2	`mean2`	MLE for mean of group 2.
3	`ratio_null`	Population ratio of means assumed for null hypothesis. `mean2 = mean1 * ratio_null`.
4	`dispersion1`	MLE for dispersion of group 1.
5	`dispersion2`	MLE for dispersion of group 2.
6	`equal_dispersion`	Were equal dispersions assumed.
7	`n1`	Sample size of group 1.
8	`n2`	Sample size of group 2.
9	`nll`	Minimum of negative log-likelihood.
10	`nparams`	Number of estimated parameters.
11	`method`	Method used for the results.
12	`mle_method`	Method used for optimization.
13	`mle_code`	Integer indicating why the optimization process terminated.
14	`mle_message`	Additional information from the optimizer.

References

Rettiganti M, Nagaraja HN (2012). "Power Analyses for Negative Binomial Models with Application to Multiple Sclerosis Clinical Trials." *Journal of Biopharmaceutical Statistics*, **22**(2), 237–259. ISSN 1054-3406, 1520-5711, doi:10.1080/10543406.2010.528105.

Aban IB, Cutter GR, Mavinga N (2009). "Inferences and power analysis concerning two negative binomial distributions with an application to MRI lesion counts data." *Computational Statistics & Data Analysis*, **53**(3), 820–833. ISSN 01679473, doi:10.1016/j.csda.2008.07.034.

See Also

sim_nb(), nll_nb

Examples

```
#-------------------------------------------------------------------------
# mle_nb() examples
#-------------------------------------------------------------------------
library(depower)

d <- sim_nb(
  n1 = 60,
  n2 = 40,
  mean1 = 10,
  ratio = 1.5,
  dispersion1 = 2,
  dispersion2 = 8
)

mle_alt <- d |>
  mle_nb_alt()

mle_null <- d |>
  mle_nb_null()

mle_alt
mle_null
```

---

nll_bnb *Negative log-likelihood for BNB*

Description

The negative log-likelihood for bivariate negative binomial outcomes.

Usage

```
nll_bnb_null(param, value1, value2, ratio_null)

nll_bnb_alt(param, value1, value2)
```

Arguments

`param`	(numeric: `(0, Inf)`) A vector of BNB parameters. Must be in the following order for each scenario: • Null: `c(mean, dispersion)` • Alternative: `c(mean1, mean2, dispersion)` for samples 1 and 2.
`value1`	(integer: `(0, Inf)`) The vector of BNB values from sample 1. Must be sorted by the subject/item index. Must not contain NAs.
`value2`	(integer: `(0, Inf)`) The vector of BNB values from sample 2. Must be sorted by the subject/item index. Must not contain NAs.
`ratio_null`	(Scalar numeric: `(0, Inf)`) The ratio of means assumed under the null hypothesis (sample 2 / sample 1). Typically `ratio_null = 1` (no difference).

Details

These functions are primarily designed for speed in simulation. Limited argument validation is performed.

Suppose $X_1 \mid G = g \sim \text{Poisson}(\mu g)$ and $X_2 \mid G = g \sim \text{Poisson}(r\mu g)$ where $G \sim \text{Gamma}(\theta, \theta^{-1})$ is the random item (subject) effect. Then $X_1, X_2 \sim \text{BNB}(\mu, r, \theta)$ is the joint distribution where X_1 and X_2 are dependent (though conditionally independent), X_1 is the count outcome for sample 1 of the items (subjects), X_2 is the count outcome for sample 2 of the items (subjects), μ is the conditional mean of sample 1, r is the ratio of the conditional means of sample 2 with respect to sample 1, and θ is the gamma distribution shape parameter which controls the dispersion and the correlation between sample 1 and 2.

The likelihood is

$$L(r, \mu, \theta \mid X_1, X_2) = \left(\frac{\theta^\theta}{\Gamma(\theta)}\right)^n \times \frac{\mu^{\sum x_{1i} + \sum x_{2i}}}{\prod_{i=1}^n x_{1i}!} \frac{r^{\sum x_{2i}}}{\prod_{i=1}^n x_{2i}!} \times \frac{\prod_{i=1}^n \Gamma(x_{1i} + x_{2i} + \theta)}{(\mu + r\mu + \theta)^{\sum (x_{1i} + x_{2i} + \theta)}}$$

and the parameter space is $\Theta = \{(r, \mu, \theta) : r, \mu, \theta > 0\}$. The log-likelihood is

$$
\begin{aligned}
l(r, \mu, \theta) = & \, n\left[\theta \ln \theta - \ln \Gamma(\theta)\right] + \\
& n(\bar{x}_1 + \bar{x}_2) \ln(\mu) + n\bar{x}_2 \ln r + \\
& \sum_{i=1}^{n} \ln \Gamma(x_{1i} + x_{2i} + \theta) - \\
& n(\bar{x}_1 + \bar{x}_2 + \theta) \ln(\mu + r\mu + \theta) - \\
& \sum_{i=1}^{n} \ln x_{1i}! - \sum_{i=1}^{n} \ln x_{2i}!
\end{aligned}
$$

Value

Scalar numeric negative log-likelihood.

References

Rettiganti M, Nagaraja HN (2012). "Power Analyses for Negative Binomial Models with Application to Multiple Sclerosis Clinical Trials." *Journal of Biopharmaceutical Statistics*, **22**(2), 237–259. ISSN 1054-3406, 1520-5711, doi:10.1080/10543406.2010.528105.

Aban IB, Cutter GR, Mavinga N (2009). "Inferences and power analysis concerning two negative binomial distributions with an application to MRI lesion counts data." *Computational Statistics & Data Analysis*, **53**(3), 820–833. ISSN 01679473, doi:10.1016/j.csda.2008.07.034.

See Also

mle_bnb

Examples

```
#-------------------------------------------------------------------------
# nll_bnb*() examples
#-------------------------------------------------------------------------
library(depower)

set.seed(1234)
d <- sim_bnb(
  n = 40,
  mean1 = 10,
  ratio = 1.2,
  dispersion = 2
)

nll_bnb_alt(
  param = c(mean1 = 10, mean2 = 12, dispersion = 2),
  value1 = d[[1L]],
  value2 = d[[2L]]
)

nll_bnb_null(
  param = c(mean = 10, dispersion = 2),
```

```
  value1 = d[[1L]],
  value2 = d[[2L]],
  ratio_null = 1
)
```

`nll_nb` *Negative log-likelihood for NB*

Description

The negative log-likelihood for two independent samples of negative binomial distributions.

Usage

```
nll_nb_null(param, value1, value2, equal_dispersion, ratio_null)

nll_nb_alt(param, value1, value2, equal_dispersion)
```

Arguments

`param`	(numeric: `(0, Inf)`) A vector of NB parameters. Must be in the following order for each scenario: • Null and unequal dispersion: `c(mean, dispersion1, dispersion2)` • Alternative and unequal dispersion: `c(mean1, mean2, dispersion1, dispersion2)` • Null and equal dispersion: `c(mean, dispersion)` • Alternative and equal dispersion: `c(mean1, mean2, dispersion)` for groups 1 and 2.
`value1`	(integer: `(0, Inf)`) The vector of NB values from group 1. Must not contain NAs.
`value2`	(integer: `(0, Inf)`) The vector of NB values from group 2. Must not contain NAs.
`equal_dispersion`	(Scalar logical) If `TRUE`, the log-likelihood is calculated assuming both groups have the same population dispersion parameter. If `FALSE` (default), the log-likelihood is calculated assuming different dispersions.
`ratio_null`	(Scalar numeric: `(0, Inf)`) The ratio of means assumed under the null hypothesis (group 2 / group 1). Typically `ratio_null = 1` (no difference).

Details

These functions are primarily designed for speed in simulation. Limited argument validation is performed.

Suppose $X_1 \sim \text{NB}(\mu, \theta_1)$ and $X_2 \sim \text{NB}(r\mu, \theta_2)$ where X_1 and X_2 are independent, X_1 is the count outcome for items in group 1, X_2 is the count outcome for items in group 2, μ is the arithmetic mean count in group 1, r is the ratio of arithmetic means for group 2 with respect to group 1, θ_1 is the dispersion parameter of group 1, and θ_2 is the dispersion parameter of group 2.

Unequal dispersion parameters:

When the dispersion parameters are not equal, the likelihood is

$$\begin{aligned} L(r, \mu, \theta_1, \theta_2 \mid X_1, X_2) = &\left(\frac{\theta_1^{\theta_1}}{\Gamma(\theta_1)}\right)^{n_1} \frac{\mu^{\sum x_{1i}}}{(\mu+\theta_1)^{\sum x_{1i}+n_1\theta_1}} \times \\ &\left(\frac{\theta_2^{\theta_2}}{\Gamma(\theta_2)}\right)^{n_2} \frac{(r\mu)^{\sum x_{2j}}}{(r\mu+\theta_2)^{\sum x_{2j}+n_2\theta_2}} \times \\ &\prod_{i=1}^{n_1} \frac{\Gamma(x_{1i}+\theta_1)}{x_{1i}!} \prod_{j=1}^{n_2} \frac{\Gamma(x_{2j}+\theta_2)}{x_{2j}!} \end{aligned}$$

and the parameter space is $\Theta = \{(r, \mu, \theta_1, \theta_2) : r, \mu, \theta_1, \theta_2 > 0\}$. The log-likelihood is

$$\begin{aligned} l(r, \mu, \theta_1, \theta_2) = &n_1 \left[\theta_1 \ln \theta_1 - \ln \Gamma(\theta_1)\right] + \\ &n_2 \left[\theta_2 \ln \theta_2 - \ln \Gamma(\theta_2)\right] + \\ &(n_1\bar{x}_1 + n_2\bar{x}_2) \ln(\mu) - n_1(\bar{x}_1 + \theta_1) \ln(\mu + \theta_1) + \\ &n_2\bar{x}_2 \ln(r) - n_2(\bar{x}_2 + \theta_2) \ln(r\mu + \theta_2) + \\ &\sum_{i=1}^{n_1} (\ln \Gamma(x_{1i} + \theta_1) - \ln(x_{1i}!)) + \\ &\sum_{j=1}^{n_2} (\ln \Gamma(x_{2j} + \theta_2) - \ln(x_{2j}!)) \end{aligned}$$

Equal dispersion parameters:

When the dispersion parameters are equal, the likelihood is

$$\begin{aligned} L(r, \mu, \theta \mid X_1, X_2) = &\left(\frac{\theta^{\theta}}{\Gamma(\theta)}\right)^{n_1+n_2} \times \\ &\frac{\mu^{\sum x_{1i}}}{(\mu+\theta)^{\sum x_{1i}+n_1\theta}} \frac{(r\mu)^{\sum x_{2j}}}{(r\mu+\theta)^{\sum x_{2j}+n_2\theta}} \times \\ &\prod_{i=1}^{n_1} \frac{\Gamma(x_{1i}+\theta)}{x_{1i}!} \prod_{j=1}^{n_2} \frac{\Gamma(x_{2j}+\theta)}{x_{2j}!} \end{aligned}$$

and the parameter space is $\Theta = \{(r, \mu, \theta) : r, \mu, \theta > 0\}$. The log-likelihood is

$$
\begin{aligned}
l(r, \mu, \theta) = {} & (n_1 + n_2)\,[\theta \ln \theta - \ln \Gamma(\theta)] + \\
& (n_1 \bar{x}_1 + n_2 \bar{x}_2) \ln(\mu) - n_1(\bar{x}_1 + \theta) \ln(\mu + \theta) + \\
& n_2 \bar{x}_2 \ln(r) - n_2(\bar{x}_2 + \theta) \ln(r\mu + \theta) + \\
& \sum_{i=1}^{n_1} (\ln \Gamma(x_{1i} + \theta) - \ln(x_{1i}!)) + \\
& \sum_{j=1}^{n_2} (\ln \Gamma(x_{2j} + \theta) - \ln(x_{2j}!))
\end{aligned}
$$

Value

Scalar numeric negative log-likelihood.

References

Rettiganti M, Nagaraja HN (2012). "Power Analyses for Negative Binomial Models with Application to Multiple Sclerosis Clinical Trials." *Journal of Biopharmaceutical Statistics*, **22**(2), 237–259. ISSN 1054-3406, 1520-5711, doi:10.1080/10543406.2010.528105.

Aban IB, Cutter GR, Mavinga N (2009). "Inferences and power analysis concerning two negative binomial distributions with an application to MRI lesion counts data." *Computational Statistics & Data Analysis*, **53**(3), 820–833. ISSN 01679473, doi:10.1016/j.csda.2008.07.034.

See Also

mle_nb

Examples

```
#-------------------------------------------------------------------------
# nll_nb_*() examples
#-------------------------------------------------------------------------
library(depower)

set.seed(1234)
d <- sim_nb(
  n1 = 60,
  n2 = 40,
  mean1 = 10,
  ratio = 1.5,
  dispersion1 = 2,
  dispersion2 = 8
)

nll_nb_alt(
  param = c(mean1 = 10, mean2 = 15, dispersion1 = 2, dispersion2 = 8),
  value1 = d[[1L]],
  value2 = d[[2L]],
  equal_dispersion = FALSE
)
```

```
nll_nb_null(
  param = c(mean = 10, dispersion1 = 2, dispersion2 = 8),
  value1 = d[[1L]],
  value2 = d[[2L]],
  equal_dispersion = FALSE,
  ratio_null = 1
)
```

`plot.depower` *Plot power objects*

Description

An automatic plot method for objects returned by `power()`.

Usage

```
## S3 method for class 'depower'
plot(
  x,
  x_axis = NULL,
  y_axis = NULL,
  color = NULL,
  facet_row = NULL,
  facet_col = NULL,
  hline = NULL,
  caption = TRUE,
  caption_width = 70L,
  ...
)
```

Arguments

`x`	(depower) The data frame returned by `power()`.
`x_axis`	(string: `NULL`; `names(x)`) The name of the column to be used for the x-axis. Automatically chosen if `NULL`.
`y_axis`	(string: `NULL`; `names(x)`) The name of the column to be used for the y-axis. Automatically chosen if `NULL`. Generally, `"power"` (default) should be used for the y-axis.
`color`	(string: `NULL`; `names(x)`) The name of the column to be used for the `ggplot2::aes()` color aesthetic. Automatically chosen if `NULL`. Use `NA` to turn off.

`facet_row` (string: `NULL`; `names(x)`)
The name of the column to be used for the `ggplot2::facet_grid()` row. Automatically chosen if `NULL`. Use `NA` to turn off.

`facet_col` (string: `NULL`; `names(x)`)
The name of the column to be used for the `ggplot2::facet_grid()` column. Automatically chosen if `NULL`. Use `NA` to turn off.

`hline` (numeric: `NULL`; `(0, 1)`)
The y-intercept at which to draw a horizontal line.

`caption` (Scalar logical: `TRUE`)
If `TRUE` (default), a caption is added to the plot. The caption includes information on parameter values that were conditioned on to generate the plot. If `FALSE`, the caption is not included.

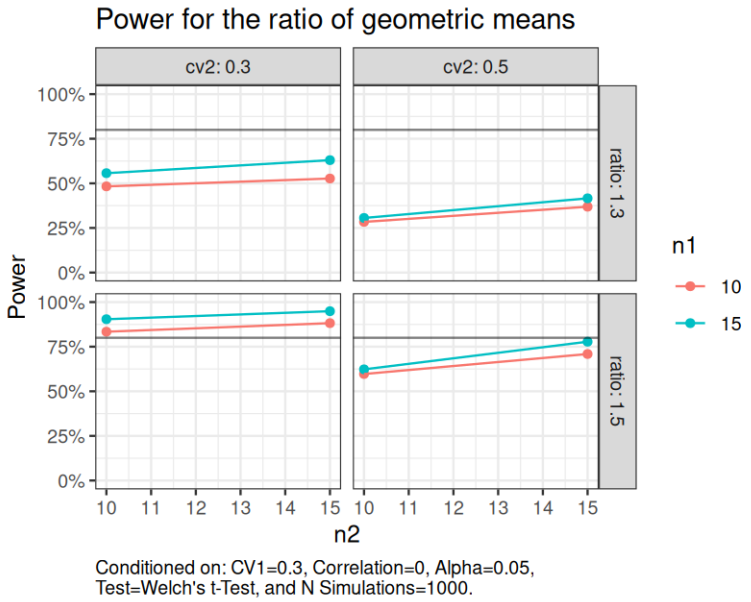
`caption_width` (Scalar integer: `70L`)
The target column number for wrapping the caption text.

`...` Unused additional arguments.

Details

If you are limited by the output from `plot.depower()`, keep in mind that the object returned by `power()` is a standard data frame. This allows you to easily plot all results with standard plotting functions. In addition, because `plot.depower()` uses ggplot2, you can modify the plot as you normally would. For example:

```
set.seed(1234)
sim_log_lognormal(
  n1 = c(10, 15),
  n2 = c(10, 15),
  ratio = c(1.3, 1.5),
  cv1 = c(0.3),
  cv2 = c(0.3, 0.5),
  nsims = 1000
) |>
  power(alpha = 0.05) |>
  plot(hline = 0.8, caption_width = 60) +
  ggplot2::theme_bw() +
  ggplot2::theme(plot.caption = ggplot2::element_text(hjust = 0)) +
  ggplot2::labs(title = "Power for the ratio of geometric means")
```



Value

A `ggplot2::ggplot()` object.

See Also

`power()`

Examples

```
#------------------------------------------------------------------------------
# plot() examples
#------------------------------------------------------------------------------
library(depower)

# Power for independent two-sample t-test
# Includes shaded region for Bayesian poster predictive interval which
# summarizes the plausible range of power estimates for a future simulation
# study based on 500 data simulations.
set.seed(1234)
sim_log_lognormal(
  n1 = c(10, 15),
  n2 = c(10, 15),
  ratio = c(1.3, 1.5),
  cv1 = c(0.3),
  cv2 = c(0.3, 0.5),
  nsims = 500
) |>
  power(alpha = 0.05) |>
  add_power_pi() |>
```

```
  plot()

# Power for dependent two-sample t-test
set.seed(1234)
sim_log_lognormal(
  n1 = c(10, 15),
  n2 = c(10, 15),
  ratio = c(1.3, 1.5),
  cv1 = c(0.3, 0.5),
  cv2 = c(0.3, 0.5),
  cor = c(0.3),
  nsims = 500
) |>
  power(alpha = 0.01) |>
  plot()

# Power for two-sample independent AND two-sample dependent t-test
set.seed(1234)
sim_log_lognormal(
  n1 = c(10, 15),
  n2 = c(10, 15),
  ratio = c(1.3, 1.5),
  cv1 = c(0.3),
  cv2 = c(0.3),
  cor = c(0, 0.3, 0.6),
  nsims = 500
) |>
  power(alpha = c(0.05, 0.01)) |>
  plot(facet_row = "cor", color = "test")

# Power for one-sample t-test
set.seed(1234)
sim_log_lognormal(
  n1 = c(10, 15),
  ratio = c(1.2, 1.4),
  cv1 = c(0.3, 0.5),
  nsims = 500
) |>
  power(alpha = c(0.05, 0.01)) |>
  plot()


# Power for independent two-sample NB test
set.seed(1234)
sim_nb(
  n1 = c(10, 15),
  mean1 = 10,
  ratio = c(1.8, 2),
  dispersion1 = 10,
  dispersion2 = 3,
  nsims = 100
) |>
  power(alpha = 0.01) |>
```

```
  plot()

# Power for BNB test
set.seed(1234)
sim_bnb(
  n = c(10, 12),
  mean1 = 10,
  ratio = c(1.3, 1.5),
  dispersion = 5,
  nsims = 100
) |>
  power(alpha = 0.01) |>
  plot()
```

---

power *Simulated power*

Description

A method to calculate power for objects returned by `sim_log_lognormal()`, `sim_nb()`, and `sim_bnb()`.

Usage

```
power(data, ..., alpha = 0.05, list_column = FALSE, ncores = 1L)
```

Arguments

`data`	(depower) The simulated data returned by `sim_log_lognormal()`, `sim_nb()`, or `sim_bnb()`. In each, argument `return_type` must be the default `"list"`.
`...`	(functions) The function(s) used to perform the test. If empty, a default test function will be selected based on `class(data)`. Names are used for labeling and should be unique. If names are empty, the call coerced to character will be used for name-value pairs. See 'Details'.
`alpha`	(numeric: `0.05`; `(0,1)`) The expected probability of rejecting a single null hypothesis when it is actually true. See 'Details'.
`list_column`	(Scalar logical: `FALSE`) If `TRUE`, the `data` and `result` list-columns are included in the returned data frame. If `FALSE` (default), the `data` and `result` list-columns are not included in the returned data frame.
`ncores`	(Scalar integer: `1L`; `[1,Inf)`) The number of cores (number of worker processes) to use. Do not set greater than the value returned by `parallel::detectCores()`. May be helpful when the number of parameter combinations is large and `nsims` is large.

Details

Power is calculated as the proportion of hypothesis tests which result in a p-value less than or equal to `alpha`. e.g.

```
sum(p <= alpha) / nsims
```

Power is defined as the expected probability of rejecting the null hypothesis for a chosen value of the unknown effect. In a multiple comparisons scenario, power is defined as the marginal power, which is the expected power of the test for each individual null hypothesis assumed to be false.

Other forms of power under the multiple comparisons scenario include disjunctive or conjunctive power. Disjunctive power is defined as the expected probability of correctly rejecting one or more null hypotheses. Conjunctive power is defined as the expected probability of correctly rejecting all null hypotheses. In the simplest case, and where all hypotheses are independent, if the marginal power is defined as π and m is the number of null hypotheses assumed to be false, then disjunctive power may be calculated as $1-(1-\pi)^m$ and conjunctive power may be calculated as π^m. Disjunctive power tends to decrease with increasingly correlated hypotheses and conjunctive power tends to increase with increasingly correlated hypotheses.

Argument `...`:

`...` are the name-value pairs for the functions used to perform the tests. If not named, the functions coerced to character will be used for the name-value pairs. Typical in non-standard evaluation, `...` accepts bare functions and converts them to a list of expressions. Each element in this list will be validated as a `call` and then evaluated on the simulated data. A `base::call()` is simply an unevaluated function. Below are some examples of specifying `...` in `power()`.

```
# Examples of specifying ... in power()
data <- sim_nb(
  n1 = 10,
  mean1 = 10,
  ratio = c(1.6, 2),
  dispersion1 = 2,
  dispersion2 = 2,
  nsims = 200
)

# ... is empty, so an appropriate default function will be provided
power(data)

# This is equivalent to leaving ... empty
power(data, "NB Wald test" = wald_test_nb())

# If not named, "wald_test_nb()" will be used to label the function
power(data, wald_test_nb())

# You can specify any parameters in the call. The data argument
# will automatically be inserted or overwritten.
data |>
  power("NB Wald test" = wald_test_nb(equal_dispersion=TRUE, link="log"))
```

```
# Multiple functions may be used.
data |>
  power(
    wald_test_nb(link='log'),
    wald_test_nb(link='sqrt'),
    wald_test_nb(link='squared'),
    wald_test_nb(link='identity')
  )

# Just like functions in a pipe, the parentheses are required.
# This will error because wald_test_nb is missing parentheses.
try(power(data, wald_test_nb))
```

In most cases*, any user created test function may be utilized in `...` if the following conditions are satisfied:

1. The function contains argument `data` which is defined as a list with the first and second elements for simulated data.
2. The return object is a list with element `p` for the p-value of the hypothesis test.

Validate with test cases beforehand.

*Simulated data of class `log_lognormal_mixed_two_sample` has both independent and dependent data. To ensure the appropriate test function is used, `power.log_lognormal_mixed_two_sample()` allows only `t_test_welch()` and `t_test_paired()` in `....` Each will be evaluated on the simulated data according to column `data$cor`. If one or both of these functions are not included in `...`, the corresponding default function will be used automatically. If any other test function is included, an error will be returned.

Argument `alpha`:

α is known as the type I assertion probability and is defined as the expected probability of rejecting a null hypothesis when it was actually true. α is compared with the p-value and used as the decision boundary for rejecting or not rejecting the null hypothesis.

The family-wise error rate is the expected probability of making one or more type I assertions among a family of hypotheses. Using Bonferroni's method, α is chosen for the family of hypotheses then divided by the number of tests performed (m). Each individual hypothesis is tested at $\frac{\alpha}{m}$. For example, if you plan to conduct 30 hypothesis tests and want to control the family-wise error rate to no greater than $\alpha = 0.05$, you would set `alpha = 0.05/30`.

The per-family error rate is the expected number of type I assertions among a family of hypotheses. If you calculate power for the scenario where you perform 1,000 hypotheses and want to control the per-family error rate to no greater than 10 type I assertions, you would choose `alpha = 10/1000`. This implicitly assumes all 1,000 hypotheses are truly null. Alternatively, if you assume 800 of these hypotheses are truly null and 200 are not, `alpha = 10/1000` would control the per-family error rate to no greater than 8 type I assertions. If it is acceptable to keep the per-family error rate as 10, setting `alpha = 10/800` would provide greater marginal power than the previous scenario.

These two methods assume that the distribution of p-values for the truly null hypotheses are uniform(0,1), but remain valid under various other testing scenarios (such as dependent tests). Other multiple comparison methods, such as FDR control, are common in practice but don't directly fit into this power simulation framework.

Column `nsims`:

The final number of valid simulations per unique set of simulation parameters may be less than the original number requested. This may occur when the test results in a missing p-value. For `wald_test_bnb()`, pathological MLE estimates, generally from small sample sizes and very small dispersions, may result in a negative estimated standard deviation of the ratio. Thus the test statistic and p-value would not be calculated. Note that simulated data from `sim_nb()` and `sim_bnb()` may also reduce `nsims` during the data simulation phase.

`nsims` denotes the effective number of simulated datasets under the alternative hypothesis, resulting in the equivalent number of hypothesis tests performed used to calculate power. If `nsims` is too small, the power estimate will have high uncertainty (wide confidence/prediction intervals). If `nsims` is too large, computation time may be prohibitive. To aid in choosing an appropriate `nsims`, functions `eval_power_ci()` and `eval_power_pi()` are helpful to understand the precision of the interval for power, before simulation takes place. Their counterparts, `add_power_ci()` and `add_power_pi()` add intervals for power to the object returned by `power()`. Functions `eval_power_ci()` and `add_power_ci()` quantify uncertainty in the true power parameter, and answer the question, "What is the plausible range of true power values given my simulation results?" Functions `eval_power_pi()` and `add_power_pi()` quantify the expected range of power estimates from a future simulation study, and answer the question, "If I run a new simulation study with m simulations, what range of power estimates might I observe?" which is particularly useful for deciding the optimal `nsims`. When the prediction intervals from `eval_power_pi()` are too wide, consider choosing a larger `nsims` before running a power simulation.

Value

A data frame with the following columns appended to the `data` object:

Name	Description
`alpha`	Type I assertion probability.
`test`	Name-value pair of the function and statistical test: `c(as.character(...) = names(...)`.
`data`	List-column of simulated data.
`result`	List-column of test results.
`power`	Power of the test for corresponding parameters.

For `power(list_column = FALSE)`, columns `data`, and `result` are excluded from the data frame.

References

Yu L, Fernandez S, Brock G (2017). "Power analysis for RNA-Seq differential expression studies." *BMC Bioinformatics*, **18**(1), 234. ISSN 1471-2105, doi:10.1186/s1285901716482.

Yu L, Fernandez S, Brock G (2020). "Power analysis for RNA-Seq differential expression studies using generalized linear mixed effects models." *BMC Bioinformatics*, **21**(1), 198. ISSN 1471-2105, doi:10.1186/s1285902035417.

Rettiganti M, Nagaraja HN (2012). "Power Analyses for Negative Binomial Models with Application to Multiple Sclerosis Clinical Trials." *Journal of Biopharmaceutical Statistics*, **22**(2), 237–259. ISSN 1054-3406, 1520-5711, doi:10.1080/10543406.2010.528105.

Aban IB, Cutter GR, Mavinga N (2009). "Inferences and power analysis concerning two negative binomial distributions with an application to MRI lesion counts data." *Computational Statistics & Data Analysis*, **53**(3), 820–833. ISSN 01679473, doi:10.1016/j.csda.2008.07.034.

Julious SA (2004). "Sample sizes for clinical trials with Normal data." *Statistics in Medicine*, **23**(12), 1921–1986. doi:10.1002/sim.1783.

Vickerstaff V, Omar RZ, Ambler G (2019). "Methods to adjust for multiple comparisons in the analysis and sample size calculation of randomised controlled trials with multiple primary outcomes." *BMC Medical Research Methodology*, **19**(1), 129. ISSN 1471-2288, doi:10.1186/s1287401907544.

See Also

`plot.depower()`, `add_power_ci()`, `add_power_pi()`, `eval_power_ci()`, `eval_power_pi()`

Examples

```
#------------------------------------------------------------------------------
# power() examples
#------------------------------------------------------------------------------
library(depower)

# Power for independent two-sample t-test
set.seed(1234)
data <- sim_log_lognormal(
  n1 = 20,
  n2 = 20,
  ratio = c(1.2, 1.4),
  cv1 = 0.4,
  cv2 = 0.4,
  cor = 0,
  nsims = 1000
)

# Welch's t-test is used by default
power(data)

# But you can specify anything else that is needed
power(
  data = data,
  "Welch's t-Test" = t_test_welch(alternative = "greater"),
  alpha = 0.01
)

# The 95% posterior predictive interval for power based on 1000 simulations
power(data) |>
  add_power_pi()

# Power for dependent two-sample t-test
set.seed(1234)
sim_log_lognormal(
  n1 = 20,
  n2 = 20,
  ratio = c(1.2, 1.4),
  cv1 = 0.4,
  cv2 = 0.4,
  cor = 0.5,
```

```
  nsims = 1000
) |>
  power()

# Power for mixed-type two-sample t-test
set.seed(1234)
sim_log_lognormal(
  n1 = 20,
  n2 = 20,
  ratio = c(1.2, 1.4),
  cv1 = 0.4,
  cv2 = 0.4,
  cor = c(0, 0.5),
  nsims = 1000
) |>
  power()

# Power for one-sample t-test
set.seed(1234)
sim_log_lognormal(
  n1 = 20,
  ratio = c(1.2, 1.4),
  cv1 = 0.4,
  nsims = 1000
) |>
  power()

# Power for independent two-sample NB test
set.seed(1234)
sim_nb(
  n1 = 10,
  mean1 = 10,
  ratio = c(1.6, 2),
  dispersion1 = 2,
  dispersion2 = 2,
  nsims = 200
) |>
  power()

# Power for BNB test
set.seed(1234)
sim_bnb(
  n = 10,
  mean1 = 10,
  ratio = c(1.4, 1.6),
  dispersion = 10,
  nsims = 200
) |>
  power()
```

`sim_bnb` *Simulate BNB data*

Description

Simulate data from the bivariate negative binomial (BNB) distribution. The BNB distribution is used to simulate count data where the event counts are jointly dependent (correlated). For independent data, see `sim_nb()`.

Usage

```
sim_bnb(
  n,
  mean1,
  mean2,
  ratio,
  dispersion,
  nsims = 1L,
  return_type = "list",
  max_zeros = 0.99
)
```

Arguments

`n`	(integer: `[2, Inf)`) The number(s) of paired observations.
`mean1`	(numeric: `(0, Inf)`) The mean(s) of sample 1 (μ_1).
`mean2`, `ratio`	(numeric: `(0, Inf)`) Only specify one of these arguments. • `mean2`: The mean(s) of sample 2 (μ_2). • `ratio`: The ratio(s) of means for sample 2 with respect to sample 1 $\left(r = \frac{\mu_2}{\mu_1}\right)$. `mean2 = ratio * mean1`
`dispersion`	(numeric: `(0, Inf)`) The gamma distribution shape parameter(s) (θ) which control the dispersion and the correlation between sample 1 and 2. See 'Details' and 'Examples'.
`nsims`	(Scalar integer: `1L`; `[1, Inf)`) The expected number of simulated data sets. If `nsims > 1`, the data is returned in a list-column of a depower simulation data frame. `nsims` may be reduced depending on `max_zeros`.
`return_type`	(string: `"list"`; `c("list", "data.frame")`) The data structure of the simulated data. If `"list"` (default), a list object is returned. If `"data.frame"` a data frame in tall format is returned. The list object provides computational efficiency and the data frame object is convenient for formulas. See 'Value'.

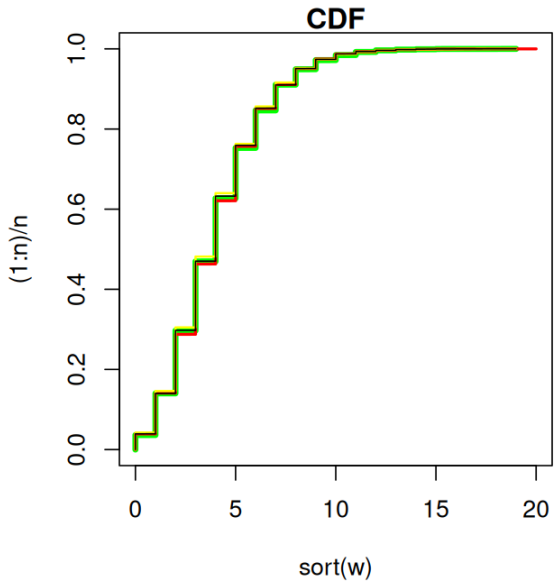
max_zeros (Scalar numeric: `0.99`; `[0, 1]`)
The maximum proportion of zeros each group in a simulated dataset is allowed to have. If the proportion of zeros is greater than this value, the corresponding data is excluded from the set of simulations. This is most likely to occur when the sample size is small and the dispersion parameter is small.

Details

The negative binomial distribution may be defined using a gamma-Poisson mixture distribution. In this case, the Poisson parameter λ is a random variable with gamma distribution. Equivalence between different parameterizations are demonstrated below:

```
# Define constants and their relationships
n <- 10000
dispersion <- 8
mu <- 4
p <- dispersion / (dispersion + mu)
q <- mu / (mu + dispersion)
variance <- mu + (mu^2 / dispersion)
rate <- p / (1 - p)
scale <- (1 - p) / p

# alternative formula for mu
mu_alt <- (dispersion * (1 - p)) / p
stopifnot(isTRUE(all.equal(mu, mu_alt)))

set.seed(20240321)

# Using built-in rnbinom with dispersion and mean
w <- rnbinom(n = n, size = dispersion, mu = mu)

# Using gamma-Poisson mixture with gamma rate parameter
x <- rpois(
  n = n,
  lambda = rgamma(n = n, shape = dispersion, rate = rate)
)

# Using gamma-Poisson mixture with gamma scale parameter
y <- rpois(
  n = n,
  lambda = rgamma(n = n, shape = dispersion, scale = scale)
)

# Using gamma-Poisson mixture with multiplicative mean and
# gamma scale parameter
z <- rpois(
  n = n,
  lambda = mu * rgamma(n = n, shape = dispersion, scale = 1/dispersion)
)
```

```
# Compare CDFs
par(mar=c(4,4,1,1))
plot(
  x = sort(w),
  y = (1:n)/n,
  xlim = range(c(w,x,y,z)),
  ylim = c(0,1),
  col = 'green',
  lwd = 4,
  type = 'l',
  main = 'CDF'
)
lines(x = sort(x), y = (1:n)/n, col = 'red', lwd = 2)
lines(x = sort(y), y = (1:n)/n, col = 'yellow', lwd = 1.5)
lines(x = sort(z), y = (1:n)/n, col = 'black')
```



The BNB distribution is implemented by compounding two conditionally independent Poisson random variables $X_1 \mid G = g \sim \text{Poisson}(\mu g)$ and $X_2 \mid G = g \sim \text{Poisson}(r\mu g)$ with a gamma random variable $G \sim \text{Gamma}(\theta, \theta^{-1})$. The probability mass function for the joint distribution of X_1, X_2 is

$$P(X_1 = x_1, X_2 = x_2) = \frac{\Gamma(x_1 + x_2 + \theta)}{(\mu + r\mu + \theta)^{x_1+x_2+\theta}} \frac{\mu^{x_1}}{x_1!} \frac{(r\mu)^{x_2}}{x_2!} \frac{\theta^\theta}{\Gamma(\theta)}$$

where $x_1, x_2 \in \mathbb{N}^{\geq 0}$ are specific values of the count outcomes, $\theta \in \mathbb{R}^{>0}$ is the `dispersion` parameter which controls the dispersion and level of correlation between the two samples (otherwise known as the shape parameter of the gamma distribution), $\mu \in \mathbb{R}^{>0}$ is the mean parameter, and $r = \frac{\mu_2}{\mu_1} \in \mathbb{R}^{>0}$ is the `ratio` parameter representing the multiplicative change in the mean of the second sample relative to the first sample. G denotes the random subject effect and the gamma

distribution scale parameter is assumed to be the inverse of the dispersion parameter (θ^{-1}) for identifiability.

Correlation decreases from 1 to 0 as the dispersion parameter increases from 0 to infinity. For a given dispersion, increasing means also increases the correlation. See 'Examples' for a demonstration.

See 'Details' in `sim_nb()` for additional information on the negative binomial distribution.

Value

If `nsims = 1` and the number of unique parameter combinations is one, the following objects are returned:

- If `return_type = "list"`, a list:

Slot	Name	Description
1		Simulated counts from sample 1.
2		Simulated counts from sample 2.

- If `return_type = "data.frame"`, a data frame:

Column	Name	Description
1	`item`	Subject/item indicator.
2	`condition`	Sample/condition indicator.
3	`value`	Simulated counts.

If `nsims > 1` or the number of unique parameter combinations is greater than one, each object described above is returned in a list-column named `data` in a depower simulation data frame:

Column	Name	Description
1	`n1`	Sample size of sample 1.
2	`n2`	Sample size of sample 2.
3	`mean1`	Mean for sample 1.
4	`mean2`	Mean for sample 2.
5	`ratio`	Ratio of means (sample 2 / sample 1).
6	`dispersion`	Gamma distribution shape parameter (dispersion).
7	`nsims`	Number of valid simulation iterations.
8	`distribution`	Distribution sampled from.
9	`data`	List-column of simulated data.

References

Yu L, Fernandez S, Brock G (2020). "Power analysis for RNA-Seq differential expression studies using generalized linear mixed effects models." *BMC Bioinformatics*, **21**(1), 198. ISSN 1471-2105, doi:10.1186/s1285902035417.

Rettiganti M, Nagaraja HN (2012). "Power Analyses for Negative Binomial Models with Application to Multiple Sclerosis Clinical Trials." *Journal of Biopharmaceutical Statistics*, **22**(2), 237–259. ISSN 1054-3406, 1520-5711, doi:10.1080/10543406.2010.528105.

Aban IB, Cutter GR, Mavinga N (2009). "Inferences and power analysis concerning two negative binomial distributions with an application to MRI lesion counts data." *Computational Statistics & Data Analysis*, **53**(3), 820–833. ISSN 01679473, doi:10.1016/j.csda.2008.07.034.

See Also

sim_nb()

Examples

```
#-------------------------------------------------------------------------
# sim_bnb() examples
#-------------------------------------------------------------------------
library(depower)

# Paired two-sample data returned in a data frame
sim_bnb(
  n = 10,
  mean1 = 10,
  ratio = 1.6,
  dispersion = 3,
  nsims = 1,
  return_type = "data.frame"
)

# Paired two-sample data returned in a list
sim_bnb(
  n = 10,
  mean1 = 10,
  ratio = 1.6,
  dispersion = 3,
  nsims = 1,
  return_type = "list"
)

# Two simulations of paired two-sample data
# returned as a list of data frames
sim_bnb(
  n = 10,
  mean1 = 10,
  ratio = 1.6,
  dispersion = 3,
  nsims = 2,
  return_type = "data.frame"
)

# Two simulations of Paired two-sample data
# returned as a list of lists
sim_bnb(
  n = 10,
  mean1 = 10,
  ratio = 1.6,
  dispersion = 3,
  nsims = 2,
  return_type = "list"
)
```

```
#------------------------------------------------------------------------------
# Visualization of the BNB distribution as dispersion varies.
# The first figure shows the marginal distribution for each group.
# The second figure shows the joint distribution for each group.
#------------------------------------------------------------------------------
set.seed(1234)
data <- lapply(
  X = c(1, 10, 100, 1000),
  FUN = function(x) {
    d <- sim_bnb(
      n = 1000,
      mean1 = 10,
      ratio = 1.5,
      dispersion = x,
      nsims = 1,
      return_type = "data.frame"
    )
    cor <- cor(
      x = d[d$condition == "1", ]$value,
      y = d[d$condition == "2", ]$value
    )
    cbind(dispersion = x, correlation = cor, d)
  }
)

data <- do.call(what = "rbind", args = data)

# Density plot of marginal distributions
ggplot2::ggplot(
  data = data,
  mapping = ggplot2::aes(x = value, fill = condition)
) +
  ggplot2::facet_wrap(
    facets = ggplot2::vars(.data$dispersion),
    ncol = 2,
    labeller = ggplot2::labeller(.rows = ggplot2::label_both)
  ) +
  ggplot2::geom_density(alpha = 0.3) +
  ggplot2::coord_cartesian(xlim = c(0, 60)) +
  ggplot2::geom_text(
    mapping = ggplot2::aes(
      x = 30,
      y = 0.12,
      label = paste0("r = ", round(correlation, 2))
    ),
    check_overlap = TRUE
  ) +
  ggplot2::labs(
    x = "Value",
    y = "Density",
    fill = "Condition",
    caption = "Mean1=10, Mean2=15, Ratio=1.5\nr=Pearson correlation"
  )
```

```
# Reshape to wide format for scatterplot
data_wide <- data.frame(
  dispersion = data[data$condition == "1", ]$dispersion,
  correlation = data[data$condition == "1", ]$correlation,
  value1 = data[data$condition == "1", ]$value,
  value2 = data[data$condition == "2", ]$value
)

# Scatterplot of joint distribution
ggplot2::ggplot(
  data = data_wide,
  mapping = ggplot2::aes(x = value1, y = value2)
) +
  ggplot2::facet_wrap(
    facets = ggplot2::vars(.data$dispersion),
    ncol = 2,
    labeller = ggplot2::labeller(.rows = ggplot2::label_both)
  ) +
  ggplot2::geom_point(alpha = 0.3) +
  ggplot2::geom_smooth(
    method = "lm",
    se = FALSE,
    color = "forestgreen"
  ) +
  ggplot2::geom_text(
    data = unique(data_wide[c("dispersion", "correlation")]),
    mapping = ggplot2::aes(
      x = 5,
      y = 55,
      label = paste0("r = ", round(correlation, 2))
    ),
    hjust = 0
  ) +
  ggplot2::coord_cartesian(xlim = c(0, 60), ylim = c(0, 60)) +
  ggplot2::labs(
    x = "Condition 1",
    y = "Condition 2",
    caption = paste0(
      "Mean1=10, Mean2=15, Ratio=1.5",
      "\nr=Pearson correlation",
      "\nSolid green line: linear regression"
    )
  )
```

---

sim_log_lognormal *Simulate log-transformed lognormal data*

Description

Simulate data from the log-transformed lognormal distribution (i.e. a normal distribution) for three scenarios:

1. One-sample data
2. Dependent two-sample data
3. Independent two-sample data

Usage

```
sim_log_lognormal(
  n1,
  n2 = NULL,
  ratio,
  cv1,
  cv2 = NULL,
  cor = 0,
  nsims = 1L,
  return_type = "list",
  messages = TRUE
)
```

Arguments

`n1`	(integer: `[2, Inf)`) The sample size(s) of sample 1.
`n2`	(integer: `NULL`; `[2, Inf)`) The sample size(s) of sample 2. Set as `NULL` if you want to simulate for the one-sample case.
`ratio`	(numeric: `(0, Inf)`) The assumed population fold change(s) of sample 2 with respect to sample 1. • For one-sample data, `ratio` is defined as the geometric mean (GM) of the original lognormal population distribution. • For dependent two-sample data, `ratio` is defined by GM(sample 2 / sample 1) of the original lognormal population distributions. – e.g. `ratio = 2` assumes that the geometric mean of all paired ratios (sample 2 / sample 1) is 2. • For independent two-sample data, the `ratio` is defined by GM(group 2) / GM(group 1) of the original lognormal population distributions. – e.g. `ratio = 2` assumes that the geometric mean of sample 2 is 2 times larger than the geometric mean of sample 1. See 'Details' for additional information.
`cv1`	(numeric: `(0, Inf)`) The coefficient of variation(s) of sample 1 in the original lognormal data.

`cv2` (numeric: `NULL`; `(0, Inf)`)
The coefficient of variation(s) of sample 2 in the original lognormal data. Set as `NULL` if you want to simulate for the one-sample case.

`cor` (numeric: `0`; `[-1, 1]`)
The correlation(s) between sample 1 and sample 2 in the original lognormal data. Not used for the one-sample case. See 'Details' for constraints based on cv1 and cv2.

`nsims` (Scalar integer: `1L`; `[1,Inf)`)
The number of simulated data sets. If `nsims > 1`, the data is returned in a list-column of a depower simulation data frame.

`return_type` (string: `"list"`; `c("list", "data.frame")`)
The data structure of the simulated data. If `"list"` (default), a list object is returned. If `"data.frame"` a data frame in tall format is returned. The list object provides computational efficiency and the data frame object is convenient for formulas. See 'Value'.

`messages` (Scalar logical: `TRUE`)
Whether or not to display messages for pathological simulation cases.

Details

Based on assumed characteristics of the original lognormal distribution, data is simulated from the corresponding log-transformed (normal) distribution. This simulated data is suitable for assessing power of a hypothesis for the geometric mean or ratio of geometric means from the original lognormal data.

This method can also be useful for other population distributions which are positive and where it makes sense to describe the ratio of geometric means. However, the lognormal distribution is theoretically correct in the sense that you can log-transform to a normal distribution, compute the summary statistic, then apply the inverse transformation to summarize on the original lognormal scale.

Notation:

Symbol	Definition
$GM(\cdot)$	Geometric mean
$AM(\cdot)$	Arithmetic mean
$CV(\cdot)$	Coefficient of variation
σ^2	Variance
ρ	Correlation
$\ln$	Natural Log
X_i	Lognormal random variable for group i
Y_i	Log-transformed lognormal random variable for group i

Fold Change and the Ratio Parameter:

The `ratio` parameter (fold change) is defined as follows for each scenario:

- One-sample: ratio $= GM(X)$
- Dependent two-sample: ratio $= GM\left(\frac{X_2}{X_1}\right)$

- Independent two-sample: ratio $= \frac{GM(X_2)}{GM(X_1)}$

For equal sample sizes of X_1 and X_2, these definitions are connected by the identity

$$\begin{aligned}\frac{GM(X_2)}{GM(X_1)} &= GM\left(\frac{X_2}{X_1}\right) \\ &= e^{AM(Y_2)-AM(Y_1)} \\ &= e^{AM(Y_2-Y_1)}.\end{aligned}$$

Coefficient of Variation:

The coefficient of variation (CV) for a random variable X is defined as

$$CV(X) = \frac{\sigma_X}{AM(X)}.$$

Relationships Between Original and Log Scales:

The following relationships allow conversion between scales.

From log scale to original lognormal scale:

$$\begin{aligned}AM(X) &= e^{AM(Y)+\sigma_Y^2/2} \\ GM(X) &= e^{AM(Y)} \\ \sigma_X^2 &= AM(X)^2\left(e^{\sigma_Y^2}-1\right) \\ CV(X) &= \frac{\sqrt{AM(X)^2\left(e^{\sigma_Y^2}-1\right)}}{AM(X)} \\ &= \sqrt{e^{\sigma_Y^2}-1}.\end{aligned}$$

From original lognormal scale to log scale:

$$\begin{aligned}AM(Y) &= \ln\left(\frac{AM(X)}{\sqrt{CV(X)^2+1}}\right) \\ \sigma_Y^2 &= \ln(CV(X)^2+1) \\ \rho_{Y_1,Y_2} &= \frac{\ln\left(\rho_{X_1,X_2}CV(X_1)CV(X_2)+1\right)}{\sqrt{\ln(CV(X_1)^2+1)}\sqrt{\ln(CV(X_2)^2+1)}} \\ &= \frac{\ln\left(\rho_{X_1,X_2}CV(X_1)CV(X_2)+1\right)}{\sigma_{Y_1}\sigma_{Y_2}}.\end{aligned}$$

Dependent Samples:

Correlation Constraints:

Not all combinations of `cor`, `cv1`, and `cv2` yield a valid correlation on the log scale. Two constraints must be satisfied.

First, the logarithm argument must be positive:

$$\rho_{X_1,X_2} \cdot CV(X_1) \cdot CV(X_2) + 1 > 0$$

which implies

$$\rho_{X_1,X_2} > \frac{-1}{CV(X_1) \cdot CV(X_2)}.$$

Second, the log-scale correlation must be in $(-1, 1)$:

$$\frac{e^{-\sigma_{Y_1}\sigma_{Y_2}} - 1}{CV(X_1) \cdot CV(X_2)} < \rho_{X_1,X_2} < \frac{e^{\sigma_{Y_1}\sigma_{Y_2}} - 1}{CV(X_1) \cdot CV(X_2)}.$$

The lower bound of the second constraint is always more restrictive than the first constraint, so the second constraint is sufficient.
When $CV(X_1) = CV(X_2) = CV$, the second constraint simplifies to:

$$\frac{-1}{CV^2 + 1} < \rho_{X_1,X_2} < 1$$

For equal CVs, only negative correlations are constrained. For example, with $CV = 0.5$, the valid range for `cor` is approximately $(-0.80, 1)$.
When $CV(X_1) \neq CV(X_2)$, both bounds may be constrained. The upper bound is strictly less than 1, meaning even positive correlations near 1 can be infeasible. For example, with $CV(X_1) = 0.1$ and $CV(X_2) = 1$, the valid range for `cor` is approximately $(-0.80, 0.87)$.

Equivalent One-Sample Representation:
Two-sample dependent data can be represented as an equivalent paired one-sample problem. First, consider the properties of variance and covariance. The variance of the difference between two dependent samples on the log scale (normal distribution) is:

$$\begin{aligned}\sigma^2_{Y_2-Y_1} &= \sigma^2_{Y_1} + \sigma^2_{Y_2} - 2Cov(Y_1, Y_2) \\ &= \sigma^2_{Y_1} + \sigma^2_{Y_2} - 2\rho_{Y_1,Y_2}\sigma_{Y_1}\sigma_{Y_2}\end{aligned}$$

Positive correlation reduces the variance of differences, while negative correlation increases it. For the special case where the two samples are uncorrelated and have equal variance: $\sigma^2_{Y_2-Y_1} = 2\sigma^2$.
Second, substitute ρ and σ^2 with their alternative forms:

$$\begin{aligned}\sigma^2_{Y_2-Y_1} &= \sigma^2_{Y_1} + \sigma^2_{Y_2} - 2\rho_{Y_1,Y_2}\sigma_{Y_1}\sigma_{Y_2} \\ &= \ln(CV(X_1)^2 + 1) + \ln(CV(X_2)^2 + 1) - 2\ln(\rho_{X_1,X_2}CV(X_1)CV(X_2) + 1) \\ &= \ln\left(\frac{(CV(X_1)^2 + 1)(CV(X_2)^2 + 1)}{(\rho_{X_1,X_2}CV(X_1)CV(X_2) + 1)^2}\right).\end{aligned}$$

Finally, denote log-transformed one-sample paired difference as Y_{diff}. It follows that $\sigma^2_{Y_{\text{diff}}} = \ln(CV(X_{\text{diff}})^2 + 1)$. We need to solve for $CV(X_{\text{diff}})$ so that the variance of the log-transformed one-sample values match the variance of the log-transformed two-sample differences. This results in

$$CV(X_{\text{diff}}) = \sqrt{\frac{(CV(X_1)^2 + 1)(CV(X_2)^2 + 1)}{(\rho_{X_1,X_2}CV(X_1)CV(X_2) + 1)^2} - 1}$$

When $CV(X_1) = CV(X_2) = CV$, this simplifies to

$$CV(X_{\text{diff}}) = \sqrt{\left(\frac{CV^2 + 1}{\rho_{X_1,X_2}CV^2 + 1}\right)^2 - 1}$$

This equivalence allows dependent two-sample data to be simulated (with lognormal scale arguments `cv1`, `cv2`, and `cor`) using a one-sample simulation (with lognormal scale argument `cv1`).

Value

If `nsims = 1` and the number of unique parameter combinations is one, the following objects are returned:

- If one-sample data with `return_type = "list"`, a list:

Slot	Name	Description
1		One sample of simulated normal values.

- If one-sample data with `return_type = "data.frame"`, a data frame:

Column	Name	Description
1	`item`	Pair/subject/item indicator.
2	`value`	Simulated normal values.

- If two-sample data with `return_type = "list"`, a list:

Slot	Name	Description
1		Simulated normal values from sample 1.
2		Simulated normal values from sample 2.

- If two-sample data with `return_type = "data.frame"`, a data frame:

Column	Name	Description
1	`item`	Pair/subject/item indicator.
2	`condition`	Time/group/condition indicator.
3	`value`	Simulated normal values.

If `nsims > 1` or the number of unique parameter combinations is greater than one, each object described above is returned in data frame, located in a list-column named `data`.

- If one-sample data, a data frame:

Column	Name	Description
1	`n1`	The sample size.
2	`ratio`	Geometric mean [GM(sample 1)].
3	`cv1`	Coefficient of variation for sample 1.
4	`nsims`	Number of data simulations.
5	`distribution`	Distribution sampled from.
6	`data`	List-column of simulated data.

- If two-sample data, a data frame:

Column	Name	Description
1	n1	Sample size of sample 1.
2	n2	Sample size of sample 2.
3	ratio	Ratio of geometric means [GM(sample 2) / GM(sample 1)] or geometric mean ratio [GM(sample
4	cv1	Coefficient of variation for sample 1.
5	cv2	Coefficient of variation for sample 2.
6	cor	Correlation between samples.
7	nsims	Number of data simulations.
8	distribution	Distribution sampled from.
9	data	List-column of simulated data.

References

Julious SA (2004). "Sample sizes for clinical trials with Normal data." *Statistics in Medicine*, **23**(12), 1921–1986. doi:10.1002/sim.1783.

Hauschke D, Steinijans VW, Diletti E, Burke M (1992). "Sample size determination for bioequivalence assessment using a multiplicative model." *Journal of Pharmacokinetics and Biopharmaceutics*, **20**(5), 557–561. ISSN 0090-466X, doi:10.1007/BF01061471.

Johnson NL, Kotz S, Balakrishnan N (1994). *Continuous univariate distributions*, Wiley series in probability and mathematical statistics, 2nd ed edition. Wiley, New York. ISBN 9780471584957 9780471584940.

Examples

```
#-------------------------------------------------------------------------
# sim_log_lognormal() examples
#-------------------------------------------------------------------------
library(depower)

# Independent two-sample data returned in a data frame
sim_log_lognormal(
  n1 = 10,
  n2 = 10,
  ratio = 1.3,
  cv1 = 0.35,
  cv2 = 0.35,
  cor = 0,
  nsims = 1,
  return_type = "data.frame"
)

# Independent two-sample data returned in a list
sim_log_lognormal(
  n1 = 10,
  n2 = 10,
  ratio = 1.3,
  cv1 = 0.35,
  cv2 = 0.35,
  cor = 0,
  nsims = 1,
```

```
  return_type = "list"
)

# Dependent two-sample data returned in a data frame
sim_log_lognormal(
  n1 = 10,
  n2 = 10,
  ratio = 1.3,
  cv1 = 0.35,
  cv2 = 0.35,
  cor = 0.4,
  nsims = 1,
  return_type = "data.frame"
)

# Dependent two-sample data returned in a list
sim_log_lognormal(
  n1 = 10,
  n2 = 10,
  ratio = 1.3,
  cv1 = 0.35,
  cv2 = 0.35,
  cor = 0.4,
  nsims = 1,
  return_type = "list"
)

# One-sample data returned in a data frame
sim_log_lognormal(
  n1 = 10,
  ratio = 1.3,
  cv1 = 0.35,
  nsims = 1,
  return_type = "data.frame"
)

# One-sample data returned in a list
sim_log_lognormal(
  n1 = 10,
  ratio = 1.3,
  cv1 = 0.35,
  nsims = 1,
  return_type = "list"
)

# Independent two-sample data: two simulations for four parameter combinations.
# Returned as a list-column of lists within a data frame
sim_log_lognormal(
  n1 = c(10, 20),
  n2 = c(10, 20),
  ratio = 1.3,
  cv1 = 0.35,
  cv2 = 0.35,
```

```
  cor = 0,
  nsims = 2,
  return_type = "list"
)

# Dependent two-sample data: two simulations for two parameter combinations.
# Returned as a list-column of lists within a data frame
sim_log_lognormal(
  n1 = c(10, 20),
  n2 = c(10, 20),
  ratio = 1.3,
  cv1 = 0.35,
  cv2 = 0.35,
  cor = 0.4,
  nsims = 2,
  return_type = "list"
)

# One-sample data: two simulations for two parameter combinations
# Returned as a list-column of lists within a data frame
sim_log_lognormal(
  n1 = c(10, 20),
  ratio = 1.3,
  cv1 = 0.35,
  nsims = 2,
  return_type = "list"
)

#-------------------------------------------------------------------------------
# Visualization of independent two-sample data from a log-transformed
# lognormal distribution with varying coefficient of variation.
#-------------------------------------------------------------------------------
cv <- expand.grid(c(0.1, 0.5, 1), c(0.1, 0.5, 1))
set.seed(1234)
data <- mapply(
  FUN = function(cv1, cv2) {
    d <- sim_log_lognormal(
      n1 = 10000,
      n2 = 10000,
      ratio = 1.5,
      cv1 = cv1,
      cv2 = cv2,
      cor = 0,
      nsims = 1,
      return_type = "data.frame"
    )
    cbind(cv1 = cv1, cv2 = cv2, d)
  },
  cv1 = cv[[1]],
  cv2 = cv[[2]],
  SIMPLIFY = FALSE
)
```

```
data <- do.call(what = "rbind", args = data)

ggplot2::ggplot(
  data = data,
  mapping = ggplot2::aes(x = value, fill = condition)
) +
  ggplot2::facet_grid(
    rows = ggplot2::vars(.data$cv2),
    cols = ggplot2::vars(.data$cv1),
    labeller = ggplot2::labeller(
      .rows = ggplot2::label_both,
      .cols = ggplot2::label_both
    )
  ) +
  ggplot2::geom_density(alpha = 0.3) +
  ggplot2::coord_cartesian(xlim = c(-3, 3)) +
  ggplot2::labs(
    x = "Value",
    y = "Density",
    fill = "Condition",
    caption = "cor=0 and ratio=1.5"
  )

#------------------------------------------------------------------------------
# Visualization of dependent two-sample data from a log-transformed lognormal
# distribution with varying correlation.
# The first figure shows the marginal distribution for each group.
# The second figure shows the joint distribution for each group.
# The third figure shows the distribution of differences.
#------------------------------------------------------------------------------
set.seed(1234)
data <- lapply(
  X = c(-0.7, -0.4, 0, 0.4, 0.7),
  FUN = function(x) {
    d <- sim_log_lognormal(
      n1 = 1000,
      n2 = 1000,
      cv1 = 0.5,
      cv2 = 0.5,
      ratio = 1.5,
      cor = x,
      nsims = 1,
      return_type = "data.frame"
    )
    cor <- cor(
      x = d[d$condition == 1, ]$value,
      y = d[d$condition == 2, ]$value
    )
    cbind(cor = x, r = cor, d)
  }
)

data <- do.call(what = "rbind", args = data)
```

```
# Density plot of marginal distributions
ggplot2::ggplot(
  data = data,
  mapping = ggplot2::aes(x = value, fill = condition)
) +
  ggplot2::facet_wrap(
    facets = ggplot2::vars(.data$cor),
    ncol = 2,
    labeller = ggplot2::labeller(.rows = ggplot2::label_both)
  ) +
  ggplot2::geom_density(alpha = 0.3) +
  ggplot2::coord_cartesian(xlim = c(-3, 3)) +
  ggplot2::geom_text(
    mapping = ggplot2::aes(
      x = -2,
      y = 0.8,
      label = paste0("r = ", round(r, 2))
    ),
    check_overlap = TRUE
  ) +
  ggplot2::labs(
    x = "Value",
    y = "Density",
    fill = "Condition",
    caption = "cv1=0.5, cv2=0.5, ratio=1.5\nr=log-scale Pearson correlation"
  )

# Reshape to wide format for scatterplot
data_wide <- data.frame(
  cor = data[data$condition == "1", ]$cor,
  r = data[data$condition == "1", ]$r,
  value1 = data[data$condition == "1", ]$value,
  value2 = data[data$condition == "2", ]$value
)

# Scatterplot of joint distribution
ggplot2::ggplot(
  data = data_wide,
  mapping = ggplot2::aes(x = value1, y = value2)
) +
  ggplot2::facet_wrap(
    facets = ggplot2::vars(.data$cor),
    ncol = 2,
    labeller = ggplot2::labeller(.rows = ggplot2::label_both)
  ) +
  ggplot2::geom_point(alpha = 0.3) +
  ggplot2::geom_smooth(
    method = "lm",
    se = FALSE,
    color = "forestgreen"
  ) +
  ggplot2::geom_text(
```

```
    data = unique(data_wide[c("cor", "r")]),
    mapping = ggplot2::aes(
      x = -2.5,
      y = 2.5,
      label = paste0("r = ", round(r, 2))
    ),
    hjust = 0
  ) +
  ggplot2::coord_cartesian(xlim = c(-3, 3), ylim = c(-3, 3)) +
  ggplot2::labs(
    x = "Condition 1",
    y = "Condition 2",
    caption = paste0(
      "cv1=0.5, cv2=0.5, ratio=1.5",
      "\nr=log-scale Pearson correlation",
      "\nSolid green line: linear regression"
    )
  )

# Density plot of differences
# Paired differences have decreasing variance as correlation increases.
data_wide$difference <- data_wide$value2 - data_wide$value1

ggplot2::ggplot(
  data = data_wide,
  mapping = ggplot2::aes(x = difference)
) +
  ggplot2::facet_wrap(
    facets = ggplot2::vars(.data$cor),
    ncol = 2,
    labeller = ggplot2::labeller(.rows = ggplot2::label_both)
  ) +
  ggplot2::geom_density(alpha = 0.3, fill = "#F8766D") +
  ggplot2::coord_cartesian(xlim = c(-3, 3)) +
  ggplot2::labs(
    x = "Difference (Condition 2 - Condition 1)",
    y = "Density",
    caption = "cv1=0.5, cv2=0.5, ratio=1.5"
  )
```

---

sim_nb *Simulate NB data*

Description

Simulate data from two independent negative binomial (NB) distributions. For paired data, see sim_bnb().

Usage

```
sim_nb(
  n1,
  n2 = n1,
  mean1,
  mean2,
  ratio,
  dispersion1,
  dispersion2 = dispersion1,
  nsims = 1L,
  return_type = "list",
  max_zeros = 0.99
)
```

Arguments

n1	(integer: [2, Inf)) The sample size(s) of group 1.
n2	(integer: n1; [2, Inf)) The sample size(s) of group 2.
mean1	(numeric: (0, Inf)) The mean(s) of group 1 (μ_1).
mean2, ratio	(numeric: (0, Inf)) Only specify one of these arguments. • mean2: The mean(s) of group 2 (μ_2). • ratio: The ratio(s) of means for group 2 with respect to group 1 $\left(r = \frac{\mu_2}{\mu_1}\right)$. mean2 = ratio * mean1
dispersion1	(numeric: (0, Inf)) The dispersion parameter(s) of group 1 (θ_1). See 'Details' and 'Examples'.
dispersion2	(numeric: dispersion1; (0, Inf)) The dispersion parameter(s) of group 2 (θ_2). See 'Details' and 'Examples'.
nsims	(Scalar integer: 1L; [1,Inf)) The expected number of simulated data sets. If nsims > 1, the data is returned in a list-column of a depower simulation data frame. nsims may be reduced depending on max_zeros.
return_type	(string: "list"; c("list", "data.frame")) The data structure of the simulated data. If "list" (default), a list object is returned. If "data.frame" a data frame in tall format is returned. The list object provides computational efficiency and the data frame object is convenient for formulas. See 'Value'.
max_zeros	(Scalar numeric: 0.99; [0, 1]) The maximum proportion of zeros each group in a simulated dataset is allowed to have. If the proportion of zeros is greater than this value, the corresponding data is excluded from the set of simulations. This is most likely to occur when the sample size is small and the dispersion parameter is small.

Details

The negative binomial distribution has many parameterizations. In the regression modeling context, it is common to specify the distribution in terms of its mean and dispersion. We use the following probability mass function:

$$\begin{aligned} P(X = x) &= \binom{x + \theta - 1}{x} \left(\frac{\theta}{\theta + \mu}\right)^{\theta} \left(\frac{\mu}{\mu + \theta}\right)^{x} \\ &= \frac{\Gamma(x + \theta)}{x!\Gamma(\theta)} \left(\frac{\theta}{\theta + \mu}\right)^{\theta} \left(\frac{\mu}{\mu + \theta}\right)^{x} \\ &= \frac{\Gamma(x + \theta)}{(\theta + \mu)^{\theta + x}} \frac{\theta^{\theta}}{\Gamma(\theta)} \frac{\mu^{x}}{x!} \end{aligned}$$

where $x \in \mathbb{N}^{\geq 0}$, $\theta \in \mathbb{R}^{>0}$ is the dispersion parameter, and $\mu \in \mathbb{R}^{>0}$ is the mean. This is analogous to the typical formulation where X is counting x failures given θ successes and $p = \frac{\theta}{\theta+\mu}$ is the probability of success on each trial. It follows that $E(X) = \mu$ and $Var(X) = \mu + \frac{\mu^2}{\theta}$. The θ parameter describes the 'dispersion' among observations. Smaller values of θ lead to overdispersion and larger values of θ decrease the overdispersion, eventually converging to the Poisson distribution.

Described above is the 'indirect quadratic parameterization' of the negative binomial distribution, which is commonly found in the R ecosystem. However, it is somewhat counterintuitive because the smaller θ gets, the larger the overdispersion. The 'direct quadratic parameterization' of the negative binomial distribution may be found in some R packages and other languages such as SAS and Stata. The direct parameterization is defined by substituting $\alpha = \frac{1}{\theta}$ ($\alpha > 0$) which results in $Var(X) = \mu + \alpha\mu^2$. In this case, the larger α gets the larger the overdispersion, and the Poisson distribution is a special case of the negative binomial distribution where $\alpha = 0$.

A general class of negative binomial models may be defined with mean μ and variance $\mu + \alpha\mu^p$. The 'linear parameterization' is then found by setting $p = 1$, resulting in $Var(X) = \mu + \alpha\mu$. It's common to label the linear parameterization as 'NB1' and the direct quadratic parameterization as 'NB2'.

See 'Details' in `sim_bnb()` for additional information on the gamma-Poisson mixture formulation of the negative binomial distribution.

Value

If `nsims = 1` and the number of unique parameter combinations is one, the following objects are returned:

- If `return_type = "list"`, a list:

Slot	Name	Description
1	`value1`	Simulated counts from group 1.
2	`value2`	Simulated counts from group 2.

- If `return_type = "data.frame"`, a data frame:

Column	Name	Description
1	`item`	Subject/item indicator.

2	condition	Group/condition indicator.
3	value	Simulated counts.

If nsims > 1 or the number of unique parameter combinations is greater than one, each object described above is returned in a list-column named data in a depower simulation data frame:

Column	Name	Description
1	n1	Sample size of group 1.
2	n2	Sample size of group 2.
3	mean1	Mean for group 1.
4	mean2	Mean for group 2.
5	ratio	Ratio of means (group 2 / group 1).
6	dispersion1	Dispersion parameter for group 1.
7	dispersion2	Dispersion parameter for group 2.
8	nsims	Number of valid simulation iterations.
9	distribution	Distribution sampled from.
10	data	List-column of simulated data.

References

Yu L, Fernandez S, Brock G (2017). "Power analysis for RNA-Seq differential expression studies." *BMC Bioinformatics*, **18**(1), 234. ISSN 1471-2105, doi:10.1186/s1285901716482.

Rettiganti M, Nagaraja HN (2012). "Power Analyses for Negative Binomial Models with Application to Multiple Sclerosis Clinical Trials." *Journal of Biopharmaceutical Statistics*, **22**(2), 237–259. ISSN 1054-3406, 1520-5711, doi:10.1080/10543406.2010.528105.

Aban IB, Cutter GR, Mavinga N (2009). "Inferences and power analysis concerning two negative binomial distributions with an application to MRI lesion counts data." *Computational Statistics & Data Analysis*, **53**(3), 820–833. ISSN 01679473, doi:10.1016/j.csda.2008.07.034.

Hilbe JM (2011). *Negative Binomial Regression*, 2 edition. Cambridge University Press. ISBN 9780521198158 9780511973420, doi:10.1017/CBO9780511973420.

Hilbe JM (2014). *Modeling count data*. Cambridge University Press, New York, NY. ISBN 9781107028333 9781107611252, doi:10.1017/CBO9781139236065.

Cameron AC, Trivedi PK (2013). *Regression Analysis of Count Data*, Econometric Society Monographs, 2 edition. Cambridge University Press. doi:10.1017/CBO9781139013567.

See Also

sim_bnb()

Examples

```
#-------------------------------------------------------------------------
# sim_nb() examples
#-------------------------------------------------------------------------
library(depower)

# Independent two-sample NB data returned in a data frame
sim_nb(
```

```
  n1 = 10,
  mean1 = 5,
  ratio = 1.6,
  dispersion1 = 0.5,
  dispersion2 = 0.5,
  nsims = 1,
  return_type = "data.frame"
)

# Independent two-sample NB data returned in a list
sim_nb(
  n1 = 10,
  mean1 = 5,
  ratio = 1.6,
  dispersion1 = 0.5,
  dispersion2 = 0.5,
  nsims = 1,
  return_type = "list"
)

# Two simulations of independent two-sample data
# returned as a list of data frames
sim_nb(
  n1 = 10,
  mean1 = 5,
  ratio = 1.6,
  dispersion1 = 0.5,
  dispersion2 = 0.5,
  nsims = 2,
  return_type = "data.frame"
)

# Two simulations of independent two-sample data
# returned as a list of lists
sim_nb(
  n1 = 10,
  mean1 = 5,
  ratio = 1.6,
  dispersion1 = 0.5,
  dispersion2 = 0.5,
  nsims = 2,
  return_type = "list"
)

#------------------------------------------------------------------------
# Visualization of the NB distribution as dispersion varies between groups.
# Small dispersion values result in higher variance (overdispersed) and
# large dispersion values result in lower variance (converges to Poisson).
#------------------------------------------------------------------------
disp <- expand.grid(c(1, 10, 100), c(1, 10, 100))
set.seed(1234)
data <- mapply(
  FUN = function(disp1, disp2) {
```

```
    d <- sim_nb(
      n1 = 1000,
      mean1 = 10,
      ratio = 1.5,
      dispersion1 = disp1,
      dispersion2 = disp2,
      nsims = 1,
      return_type = "data.frame"
    )
    cbind(dispersion1 = disp1, dispersion2 = disp2, d)
  },
  disp1 = disp[[1]],
  disp2 = disp[[2]],
  SIMPLIFY = FALSE
)

data <- do.call(what = "rbind", args = data)

ggplot2::ggplot(
  data = data,
  mapping = ggplot2::aes(x = value, fill = condition)
) +
  ggplot2::facet_grid(
    rows = ggplot2::vars(.data$dispersion2),
    cols = ggplot2::vars(.data$dispersion1),
    labeller = ggplot2::labeller(
      .rows = ggplot2::label_both,
      .cols = ggplot2::label_both
    )
  ) +
  ggplot2::geom_density(alpha = 0.3) +
  ggplot2::coord_cartesian(xlim = c(0, 50)) +
  ggplot2::labs(
    x = "Value",
    y = "Density",
    fill = "Condition",
    caption = "Mean1=10, Mean2=15, ratio=1.5"
  )
```

`t_test_paired` *Paired and one-sample t-Tests*

Description

Performs paired and one-sample t-Tests.

Usage

```
t_test_paired(data, alternative = "two.sided", ci_level = NULL, mean_null = 0)
```

Arguments

`data`	(list) A list whose first element is the vector of normal values from sample 1 and the second element is the vector of normal values from sample 2. Both vectors must be the same sample size and sorted by the subject/item index. If `length(data) == 1L`, the one-sample test is used. NAs are silently excluded. The default output from `sim_log_lognormal()`.
`alternative`	(string: `"two.sided"`) The alternative hypothesis. Must be one of `"two.sided"`, `"greater"`, or `"less"`. See 'Details' for additional information.
`ci_level`	(Scalar numeric: `NULL`; `(0, 1)`) If `NULL`, confidence intervals are set as `NA`. If in `(0, 1)`, confidence intervals are calculated at the specified level.
`mean_null`	(Scalar numeric: `0`; `(-Inf, Inf)`) The mean or mean difference assumed under the null hypothesis. See 'Details' for additional information.

Details

This function is primarily designed for speed in simulation. Missing values are silently excluded.

The one-sample test is used for both the true one-sample scenario and for the paired differences from a dependent two-sample scenario. Below we use paired difference language as that is the most common case. The hypotheses for the paired t-test are

$$H_{null} : \mu_{diff} = \mu_{null}$$

$$H_{alt} : \begin{cases} \mu_{diff} \neq \mu_{null} & \text{two-sided} \\ \mu_{diff} > \mu_{null} & \text{greater than} \\ \mu_{diff} < \mu_{null} & \text{less than} \end{cases}$$

where $\mu_{diff} = AM(X_2 - X_1)$ is the arithmetic mean of the paired differences (sample 2 - sample 1) and μ_{null} is a constant for the assumed population mean difference (usually $\mu_{null} = 0$).

The test statistic is

$$T = \frac{\bar{x}_{diff} - \mu_{null}}{\sqrt{\frac{s^2}{n}}}$$

where $\bar{x}_{diff}$ is the sample mean of the differences, μ_{null} is the population mean difference assumed under the null hypothesis, n is the sample size of the differences, and s^2 is the sample variance.

The critical value of the test statistic has degrees of freedom

$$df = n - 1$$

and the p-value is calculated as

$$p = \begin{cases} 2\min\{P(T \geq t_{n-1} \mid H_{null}), P(T \leq t_{n-1} \mid H_{null})\} & \text{two-sided} \\ P(T \geq t_{n-1} \mid H_{null}) & \text{greater than} \\ P(T \leq t_{n-1} \mid H_{null}) & \text{less than} \end{cases}$$

Let $GM(\cdot)$ be the geometric mean and $AM(\cdot)$ be the arithmetic mean. For dependent lognormal samples X_1 and X_2 it follows that $\ln X_1$ and $\ln X_2$ are dependent normally distributed variables. Setting $\mu_{diff} = AM(\ln X_2 - \ln X_1)$ we have

$$e^{\mu_{diff}} = GM\left(\frac{X_2}{X_1}\right)$$

This forms the basis for making inference about the geometric mean ratio of the original lognormal data using the mean difference of the log transformed normal data.

Value

A list with the following elements:

Slot	Subslot	Name	Description
1		`t`	Value of the t-statistic.
2		`df`	Degrees of freedom for the t-statistic.
3		`p`	p-value.
4		`mean_diff`	Estimated mean or mean of the differences (sample 2 – sample 1).
4	1	`estimate`	Point estimate.
4	2	`lower`	Confidence interval lower bound.
4	3	`upper`	Confidence interval upper bound.
5		`n`	Number of paired observations.
6		`method`	Method used for the results.
7		`alternative`	The alternative hypothesis.
8		`ci_level`	The confidence level.
9		`mean_null`	Assumed population mean of the differences under the null hypothesis.

References

Julious SA (2004). "Sample sizes for clinical trials with Normal data." *Statistics in Medicine*, **23**(12), 1921–1986. doi:10.1002/sim.1783.

Hauschke D, Steinijans VW, Diletti E, Burke M (1992). "Sample size determination for bioequivalence assessment using a multiplicative model." *Journal of Pharmacokinetics and Biopharmaceutics*, **20**(5), 557–561. ISSN 0090-466X, doi:10.1007/BF01061471.

Johnson NL, Kotz S, Balakrishnan N (1994). *Continuous univariate distributions*, Wiley series in probability and mathematical statistics, 2nd ed edition. Wiley, New York. ISBN 9780471584957 9780471584940.

See Also

`t_test_welch()`

Examples

```
#------------------------------------------------------------------------
# t_test_paired() examples
#------------------------------------------------------------------------
library(depower)

# One-sample t-test
set.seed(1234)
t_test1 <- sim_log_lognormal(
  n1 = 40,
  ratio = 1.5,
  cv1 = 0.4
) |>
  t_test_paired(ci_level = 0.95)

t_test1

# Paired t-test using two dependent samples
set.seed(1234)
t_test2 <- sim_log_lognormal(
  n1 = 40,
  n2 = 40,
  ratio = 1.5,
  cv1 = 0.4,
  cv2 = 0.2,
  cor = 0.3
) |>
  t_test_paired(ci_level = 0.95)

t_test2
```

---

t_test_welch *Welch's t-Test*

Description

Performs Welch's independent two-sample t-test.

Usage

```
t_test_welch(data, alternative = "two.sided", ci_level = NULL, mean_null = 0)
```

Arguments

data	(list) A list whose first element is the vector of normal values from group 1 and the second element is the vector of normal values from group 2. NAs are silently excluded. The default output from `sim_log_lognormal()`.

`alternative` (string: `"two.sided"`)
The alternative hypothesis. Must be one of `"two.sided"`, `"greater"`, or `"less"`. See 'Details' for additional information.

`ci_level` (Scalar numeric: `NULL`; `(0, 1)`)
If `NULL`, confidence intervals are set as `NA`. If in `(0, 1)`, confidence intervals are calculated at the specified level.

`mean_null` (Scalar numeric: `0`; `(-Inf, Inf)`)
The difference of means assumed under the null hypothesis. See 'Details' for additional information.

Details

This function is primarily designed for speed in simulation. Missing values are silently excluded.

The hypotheses for Welch's independent two-sample t-test are

$$H_{null} : \mu_2 - \mu_1 = \mu_{null}$$
$$H_{alt} : \begin{cases} \mu_2 - \mu_1 \neq \mu_{null} & \text{two-sided} \\ \mu_2 - \mu_1 > \mu_{null} & \text{greater than} \\ \mu_2 - \mu_1 < \mu_{null} & \text{less than} \end{cases}$$

where μ_1 is the population mean of group 1, μ_2 is the population mean of group 2, and μ_{null} is a constant for the assumed difference of population means (usually $\mu_{null} = 0$).

The test statistic is

$$T = \frac{(\bar{x}_2 - \bar{x}_1) - \mu_{null}}{\sqrt{\frac{s_1^2}{n_1} + \frac{s_2^2}{n_2}}}$$

where $\bar{x}_1$ and $\bar{x}_2$ are the sample means, μ_{null} is the difference of population means assumed under the null hypothesis, n_1 and n_2 are the sample sizes, and s_1^2 and s_2^2 are the sample variances.

The critical value of the test statistic uses the Welch–Satterthwaite degrees of freedom

$$v = \frac{\left(\frac{s_1^2}{n_1} + \frac{s_2^2}{n_2}\right)^2}{(N_1 - 1)^{-1}\left(\frac{s_1^2}{n_1}\right)^2 + (N_2 - 1)^{-1}\left(\frac{s_2^2}{n_2}\right)^2}$$

and the p-value is calculated as

$$p = \begin{cases} 2\min\{P(T \geq t_v \mid H_{null}), P(T \leq t_v \mid H_{null})\} & \text{two-sided} \\ P(T \geq t_v \mid H_{null}) & \text{greater than} \\ P(T \leq t_v \mid H_{null}) & \text{less than} \end{cases}$$

Let $GM(\cdot)$ be the geometric mean and $AM(\cdot)$ be the arithmetic mean. For independent lognormal variables X_1 and X_2 it follows that $\ln X_1$ and $\ln X_2$ are independent normally distributed variables. Defining $\mu_{X_2} - \mu_{X_1} = AM(\ln X_2) - AM(\ln X_1)$ we have

$$e^{\mu_{X_2}-\mu_{X_1}} = \frac{GM(X_2)}{GM(X_1)}$$

This forms the basis for making inference about the ratio of geometric means of the original log-normal data using the difference of means of the log transformed normal data.

Value

A list with the following elements:

Slot	Subslot	Name	Description
1		`t`	Value of the t-statistic.
2		`df`	Degrees of freedom for the t-statistic.
3		`p`	p-value.
4		`diff_mean`	Estimated difference of means (group 2 – group 1).
4	1	`estimate`	Point estimate.
4	2	`lower`	Confidence interval lower bound.
4	3	`upper`	Confidence interval upper bound.
5		`mean1`	Estimated mean of group 1.
6		`mean2`	Estimated mean of group 2.
7		`n1`	Sample size of group 1.
8		`n2`	Sample size of group 2.
9		`method`	Method used for the results.
10		`alternative`	The alternative hypothesis.
11		`ci_level`	The confidence level.
12		`mean_null`	Assumed population difference of the means under the null hypothesis.

References

Julious SA (2004). "Sample sizes for clinical trials with Normal data." *Statistics in Medicine*, **23**(12), 1921–1986. doi:10.1002/sim.1783.

Hauschke D, Steinijans VW, Diletti E, Burke M (1992). "Sample size determination for bioequivalence assessment using a multiplicative model." *Journal of Pharmacokinetics and Biopharmaceutics*, **20**(5), 557–561. ISSN 0090-466X, doi:10.1007/BF01061471.

Johnson NL, Kotz S, Balakrishnan N (1994). *Continuous univariate distributions*, Wiley series in probability and mathematical statistics, 2nd ed edition. Wiley, New York. ISBN 9780471584957 9780471584940.

See Also

`t_test_paired()`

Examples

```
#------------------------------------------------------------------------------
# t_test_welch() examples
#------------------------------------------------------------------------------
library(depower)
```

```
# Welch's t-test
set.seed(1234)
sim_log_lognormal(
  n1 = 40,
  n2 = 40,
  ratio = 1.5,
  cv1 = 0.4,
  cv2 = 0.4
) |>
  t_test_welch(ci_level = 0.95)
```

`wald_test_bnb` *Wald test for BNB ratio of means*

Description

Wald test for the ratio of means from bivariate negative binomial outcomes.

Usage

```
wald_test_bnb(
  data,
  ci_level = NULL,
  link = "log",
  ratio_null = 1,
  distribution = asymptotic(),
  ...
)
```

Arguments

`data`	(list) A list whose first element is the vector of negative binomial values from sample 1 and the second element is the vector of negative binomial values from sample 2. Each vector must be sorted by the subject/item index and must be the same sample size. NAs are silently excluded. The default output from `sim_bnb()`.
`ci_level`	(Scalar numeric: `NULL`; `(0, 1)`) If `NULL`, confidence intervals are set as `NA`. If in `(0, 1)`, confidence intervals are calculated at the specified level.
`link`	(Scalar string: `"log"`) The one-to-one link function for transformation of the ratio in the test hypotheses. Must be one of `"log"` (default), `"sqrt"`, `"squared"`, or `"identity"`. See 'Details' for additional information.
`ratio_null`	(Scalar numeric: `1`; `(0, Inf)`) The (pre-transformation) ratio of means assumed under the null hypothesis (sample 2 / sample 1). Typically `ratio_null = 1` (no difference). See 'Details' for additional information.

`distribution` (function: `asymptotic()` or `simulated()`)
The method used to define the distribution of the χ^2 Wald test statistic under the null hypothesis. See 'Details' and `asymptotic()` or `simulated()` for additional information.

`...` Optional arguments passed to the MLE function `mle_bnb()`.

Details

This function is primarily designed for speed in simulation. Missing values are silently excluded.

Suppose $X_1 \mid G = g \sim \text{Poisson}(\mu g)$ and $X_2 \mid G = g \sim \text{Poisson}(r\mu g)$ where $G \sim \text{Gamma}(\theta, \theta^{-1})$ is the random item (subject) effect. Then $X_1, X_2 \sim \text{BNB}(\mu, r, \theta)$ is the joint distribution where X_1 and X_2 are dependent (though conditionally independent), X_1 is the count outcome for sample 1 of the items (subjects), X_2 is the count outcome for sample 2 of the items (subjects), μ is the conditional mean of sample 1, r is the ratio of the conditional means of sample 2 with respect to sample 1, and θ is the gamma distribution shape parameter which controls the dispersion and the correlation between sample 1 and 2.

The hypotheses for the Wald test of r are

$$H_{null} : f(r) = f(r_{null})$$
$$H_{alt} : f(r) \neq f(r_{null})$$

where $f(\cdot)$ is a one-to-one link function with nonzero derivative, $r = \frac{\bar{X}_2}{\bar{X}_1}$ is the population ratio of arithmetic means for sample 2 with respect to sample 1, and r_{null} is a constant for the assumed null population ratio of means (typically $r_{null} = 1$).

Rettiganti and Nagaraja (2012) found that $f(r) = r^2$, $f(r) = r$, and $f(r) = r^{0.5}$ had greatest power when $r < 1$. However, when $r > 1$, $f(r) = \ln r$, the likelihood ratio test, and $f(r) = r^{0.5}$ had greatest power. $f(r) = r^2$ was biased when $r > 1$. Both $f(r) = \ln r$ and $f(r) = r^{0.5}$ produced acceptable results for any r value. These results depend on the use of asymptotic vs. exact critical values.

The Wald test statistic is

$$W(f(\hat{r})) = \left(\frac{f\left(\frac{\bar{x}_2}{\bar{x}_1}\right) - f(r_{null})}{f'(\hat{r})\hat{\sigma}_{\hat{r}}} \right)^2$$

where

$$\hat{\sigma}^2_{\hat{r}} = \frac{\hat{r}(1+\hat{r})(\hat{\mu} + \hat{r}\hat{\mu} + \hat{\theta})}{n\left[\hat{\mu}(1+\hat{r})(\hat{\mu}+\hat{\theta}) - \hat{\theta}\hat{r}\right]}$$

Under H_{null}, the Wald test statistic is asymptotically distributed as χ^2_1. The approximate level α test rejects H_{null} if $W(f(\hat{r})) \geq \chi^2_1(1-\alpha)$. However, the asymptotic critical value is known to underestimate the exact critical value and the nominal significance level may not be achieved for small sample sizes. The level of significance inflation also depends on $f(\cdot)$ and is most severe for $f(r) = r^2$ where only the exact critical value should be used. Argument `distribution` allows control of the distribution of the χ^2_1 test statistic under the null hypothesis by use of functions `asymptotic()` and `simulated()`.

Value

A list with the following elements:

Slot	Subslot	Name	Description
1		`chisq`	χ^2 test statistic for the ratio of means.
2		`df`	Degrees of freedom.
3		`p`	p-value.
4		`ratio`	Estimated ratio of means (group 2 / group 1).
4	1	`estimate`	Point estimate.
4	2	`lower`	Confidence interval lower bound.
4	3	`upper`	Confidence interval upper bound.
5		`mean1`	Estimated mean of sample 1.
6		`mean2`	Estimated mean of sample 2.
7		`dispersion`	Estimated dispersion.
8		`n1`	The sample size of sample 1.
9		`n2`	The sample size of sample 2.
10		`method`	Method used for the results.
11		`ci_level`	The confidence level.
12		`link`	Link function used to transform the ratio of means in the test hypotheses.
13		`ratio_null`	Assumed ratio of means under the null hypothesis.
14		`mle_code`	Integer indicating why the optimization process terminated.
15		`mle_message`	Information from the optimizer.

References

Rettiganti M, Nagaraja HN (2012). "Power Analyses for Negative Binomial Models with Application to Multiple Sclerosis Clinical Trials." *Journal of Biopharmaceutical Statistics*, **22**(2), 237–259. ISSN 1054-3406, 1520-5711, doi:10.1080/10543406.2010.528105.

Aban IB, Cutter GR, Mavinga N (2009). "Inferences and power analysis concerning two negative binomial distributions with an application to MRI lesion counts data." *Computational Statistics & Data Analysis*, **53**(3), 820–833. ISSN 01679473, doi:10.1016/j.csda.2008.07.034.

See Also

`lrt_bnb()`

Examples

```
#-------------------------------------------------------------------------
# wald_test_bnb() examples
#-------------------------------------------------------------------------
library(depower)

set.seed(1234)
sim_bnb(
  n = 40,
  mean1 = 10,
  ratio = 1.2,
  dispersion = 2
```

```
) |>
  wald_test_bnb()
```

`wald_test_nb` *Wald test for NB ratio of means*

Description

Wald test for the ratio of means from two independent negative binomial outcomes.

Usage

```
wald_test_nb(
  data,
  equal_dispersion = FALSE,
  ci_level = NULL,
  link = "log",
  ratio_null = 1,
  distribution = asymptotic(),
  ...
)
```

Arguments

`data`	(list) A list whose first element is the vector of negative binomial values from group 1 and the second element is the vector of negative binomial values from group 2. NAs are silently excluded. The default output from `sim_nb()`.
`equal_dispersion`	(Scalar logical: `FALSE`) If `TRUE`, the Wald test is calculated assuming both groups have the same population dispersion parameter. If `FALSE` (default), the Wald test is calculated assuming different dispersions.
`ci_level`	(Scalar numeric: `NULL`; `(0, 1)`) If `NULL`, confidence intervals are set as `NA`. If in `(0, 1)`, confidence intervals are calculated at the specified level.
`link`	(Scalar string: `"log"`) The one-to-one link function for transformation of the ratio in the test hypotheses. Must be one of `"log"` (default), `"sqrt"`, `"squared"`, or `"identity"`. See 'Details' for additional information.
`ratio_null`	(Scalar numeric: `1`; `(0, Inf)`) The (pre-transformation) ratio of means assumed under the null hypothesis (group 2 / group 1). Typically `ratio_null = 1` (no difference). See 'Details' for additional information.

`distribution` (function: `asymptotic()` or `simulated()`)
The method used to define the distribution of the χ^2 Wald test statistic under the null hypothesis. See 'Details' and `asymptotic()` or `simulated()` for additional information.

`...` Optional arguments passed to the MLE function `mle_nb()`.

Details

This function is primarily designed for speed in simulation. Missing values are silently excluded.

Suppose $X_1 \sim NB(\mu, \theta_1)$ and $X_2 \sim NB(r\mu, \theta_2)$ where X_1 and X_2 are independent, X_1 is the count outcome for items in group 1, X_2 is the count outcome for items in group 2, μ is the arithmetic mean count in group 1, r is the ratio of arithmetic means for group 2 with respect to group 1, θ_1 is the dispersion parameter of group 1, and θ_2 is the dispersion parameter of group 2.

The hypotheses for the Wald test of r are

$$H_{null} : f(r) = f(r_{null})$$
$$H_{alt} : f(r) \neq f(r_{null})$$

where $f(\cdot)$ is a one-to-one link function with nonzero derivative, $r = \frac{\bar{X}_2}{\bar{X}_1}$ is the population ratio of arithmetic means for group 2 with respect to group 1, and r_{null} is a constant for the assumed null population ratio of means (typically $r_{null} = 1$).

Rettiganti and Nagaraja (2012) found that $f(r) = r^2$ and $f(r) = r$ had greatest power when $r < 1$. However, when $r > 1$, $f(r) = \ln r$, the likelihood ratio test, and the Rao score test have greatest power. Note that $f(r) = \ln r$, LRT, and RST were unbiased tests while the $f(r) = r$ and $f(r) = r^2$ tests were biased when $r > 1$. The $f(r) = \ln r$, LRT, and RST produced acceptable results for any r value. These results depend on the use of asymptotic vs. exact critical values.

The Wald test statistic is

$$W(f(\hat{r})) = \left(\frac{f\left(\frac{\bar{x}_2}{\bar{x}_1}\right) - f(r_{null})}{f'(\hat{r})\hat{\sigma}_{\hat{r}}} \right)^2$$

where

$$\hat{\sigma}^2_{\hat{r}} = \frac{\hat{r}\left[n_1\hat{\theta}_1(\hat{r}\hat{\mu} + \hat{\theta}_2) + n_2\hat{\theta}_2\hat{r}(\hat{\mu} + \hat{\theta}_1)\right]}{n_1 n_2 \hat{\theta}_1 \hat{\theta}_2 \hat{\mu}}$$

Under H_{null}, the Wald test statistic is asymptotically distributed as χ^2_1. The approximate level α test rejects H_{null} if $W(f(\hat{r})) \geq \chi^2_1(1-\alpha)$. However, the asymptotic critical value is known to underestimate the exact critical value and the nominal significance level may not be achieved for small sample sizes. The level of significance inflation also depends on $f(\cdot)$ and is most severe for $f(r) = r^2$ where only the exact critical value should be used. Argument `distribution` allows control of the distribution of the χ^2_1 test statistic under the null hypothesis by use of functions `asymptotic()` and `simulated()`.

Note that standalone use of this function with `equal_dispersion = FALSE` and `distribution = simulated()`, e.g.

```
data |>
  wald_test_nb(
    equal_dispersion = FALSE,
    distribution = simulated()
  )
```

results in a nonparametric randomization test based on label permutation. This violates the assumption of exchangeability for the randomization test because the labels are not exchangeable when the null hypothesis assumes unequal dispersions. However, used inside `power()`, e.g.

```
data |>
  power(
    wald_test_nb(
      equal_dispersion = FALSE,
      distribution = simulated()
    )
  )
```

results in parametric resampling and no label permutation in performed. Thus, setting `equal_dispersion = FALSE` and `distribution = simulated()` is only recommended when `wald_test_nb()` is used inside of `power()`. See also, `simulated()`.

Value

A list with the following elements:

Slot	Subslot	Name	Description
1		`chisq`	χ^2 test statistic for the ratio of means.
2		`df`	Degrees of freedom.
3		`p`	p-value.
4		`ratio`	Estimated ratio of means (group 2 / group 1).
4	1	`estimate`	Point estimate.
4	2	`lower`	Confidence interval lower bound.
4	3	`upper`	Confidence interval upper bound.
5		`mean1`	Estimated mean of group 1.
6		`mean2`	Estimated mean of group 2.
7		`dispersion1`	Estimated dispersion of group 1.
8		`dispersion2`	Estimated dispersion of group 2.
9		`n1`	Sample size of group 1.
10		`n2`	Sample size of group 2.
11		`method`	Method used for the results.
12		`ci_level`	The confidence level.
13		`equal_dispersion`	Whether or not equal dispersions were assumed.
14		`link`	Link function used to transform the ratio of means in the test hypotheses.
15		`ratio_null`	Assumed ratio of means under the null hypothesis.
16		`mle_code`	Integer indicating why the optimization process terminated.
17		`mle_message`	Information from the optimizer.

References

Rettiganti M, Nagaraja HN (2012). "Power Analyses for Negative Binomial Models with Application to Multiple Sclerosis Clinical Trials." *Journal of Biopharmaceutical Statistics*, **22**(2), 237–259. ISSN 1054-3406, 1520-5711, doi:10.1080/10543406.2010.528105.

Aban IB, Cutter GR, Mavinga N (2009). "Inferences and power analysis concerning two negative binomial distributions with an application to MRI lesion counts data." *Computational Statistics & Data Analysis*, **53**(3), 820–833. ISSN 01679473, doi:10.1016/j.csda.2008.07.034.

See Also

`lrt_nb()`

Examples

```
#-------------------------------------------------------------------------
# wald_test_nb() examples
#-------------------------------------------------------------------------
library(depower)

set.seed(1234)
sim_nb(
  n1 = 60,
  n2 = 40,
  mean1 = 10,
  ratio = 1.5,
  dispersion1 = 2,
  dispersion2 = 8
) |>
  wald_test_nb()
```

Index