Package ‘PASSED’

July 21, 2020

Title Calculate Power and Sample Size for Two Sample Mean Tests

Version 1.0-3

Date 2020-07-21

Description Power calculations are a critical component of any research study to determine the minimum sample size necessary to detect differences between multiple groups. Here we present an 'R' package, 'PASSED', that performs power and sample size calculations for the test of two-sample means or ratios with data following beta, gamma (Chang et al. (2011), <doi:10.1007/s00180-010-0209-1>), normal, Poisson (Gu et al. (2008), <doi:10.1002/bimj.200710403>), binomial, geometric, and negative binomial (Zhu and Lakkis (2014), <doi:10.1002/sim.5947>) distributions.

Depends R (>= 3.6)

License GPL (>= 2)

Encoding UTF-8

LazyData true

RoxygenNote 7.1.0

Imports betareg, stats, rootSolve

Suggests knitr, rmarkdown, MKmisc

NeedsCompilation no

Author Jinpu Li [aut, cre],
Ryan Knigge [aut],
Emily Leary [aut]

Maintainer Jinpu Li <lijinp@health.missouri.edu>

Repository CRAN

Date/Publication 2020-07-21 20:40:13 UTC

R topics documented:

power_Beta 2
power_Binomial 3
power_Gamma 4
power_Geometric 5

power_NegativeBinomial 6
power_Normal 8
power_Poisson 9

Index **11**

---

power_Beta *Power Calculations for Test of Two Beta Means*

Description

Compute the power for a test of two sample means with beta distributions, or determine the minimum sample size to obtain a target power.

Usage

```
power_Beta(n1 = NULL, n2 = NULL, power = NULL, sig.level = 0.05,
mu1 = NULL, sd1 = NULL, mu2 = NULL, equal.sample = TRUE,
trials = 100, equal.precision = TRUE, sd2 = NULL,
link.type = c("logit", "probit", "cloglog", "cauchit", "log", "loglog"))
```

Arguments

`n1`	sample size in group 1, or sample size in each group if `equal.sample = TRUE`
`n2`	sample size in group 2
`power`	power of test (1 minus Type II error probability)
`sig.level`	significance level (Type I error probability)
`mu1`	sample mean of group 1
`sd1`	standard deviation for group 1
`mu2`	sample mean of group 2
`equal.sample`	equal sample sizes for two groups, see details
`trials`	number of trials in simulation
`equal.precision`	equal dispersion parameter assumption in simulation
`sd2`	standard deviation for group 2. Only applicable when `equal.precision = FALSE`
`link.type`	type of link used in the beta regression, see details

Details

Exactly one of the parameters `n1`, `n2` and `power` must be passed as NULL, and that parameter is determined from the others.

This function allows you to set the number of trials in the simulation to control the result accuracy, and type of link used in the beta regression. You can choose one of the following: "logit", "probit", "cloglog", "cauchit", "log", "loglog".

Value

Object of class "power.htest", a list of the arguments (including the computed one) augmented with method and note elements.

Examples

```
# calculate power
power_Beta(mu1 = 0.5, mu2 = 0.80, sd1 = 0.25, n1 = 60)
# calculate sample size for both groups
power_Beta(mu1 = 0.5, mu2 = 0.80, sd1 = 0.25, power=0.8)
```

---

power_Binomial *Power Calculations for Two-Sample Test for Proportions*

Description

Compute power of test, or determine parameters to obtain target power for equal and unequal sample sizes.

Usage

```
power_Binomial(n1 = NULL, n2 = NULL, power = NULL, sig.level = 0.05,
p1 = 0.5, p2 = 0.5, equal.sample = TRUE, alternative = c("two.sided", "one.sided"))
```

Arguments

`n1`	sample size in group 1, or sample size in each group if equal.sample = TRUE
`n2`	sample size in group 2
`power`	power of test (1 minus Type II error probability)
`sig.level`	significance level (Type I error probability)
`p1`	probability in group 1
`p2`	probability in group 2
`equal.sample`	equal sample sizes for two groups, see details
`alternative`	one- or two-sided test

Details

Exactly one of the parameters `n1`, `n2`, `p1`, `p2`, `power`, and `sig.level` must be passed as NULL, and that parameter is determined from the others. Notice that `p1`, `p2`, `sig.level` have non-NULL defaults, so NULL must be explicitly expressed if you want to compute them.

If `equal.sample = TRUE` is used, N in output will denote the number in each group.

Value

Object of class "power.htest", a list of the arguments (including the computed one) augmented with note and method elements.

Examples

```
# calculate power, equal sizes
power_Binomial(n1 = 100, p1 = 0.5, p2 = 0.7)
# calculate power, unequal sizes
power_Binomial(n1 = 150, n2 = 100, p1 = 0.5, p2 = 0.7)
# calculate n2
power_Binomial(n1 = 100, p1 = 0.5, p2 = 0.7, power = 0.9, equal.sample = FALSE)
```

---

power_Gamma *Power Calculations for Test of Two Gamma Means*

Description

Compute the power for a test of two sample means with Gamma distributions, or determine parameters to obtain a target power.

Usage

```
power_Gamma(n1 = NULL, n2 = NULL, power = NULL, sig.level = 0.05,
mu1 = NULL, mu2 = NULL, gmu1 = NULL, gmu2 = NULL,
trials = 100, M = 10000, equal.sample = TRUE, equal.shape = NULL)
```

Arguments

`n1`	sample size in group 1, or sample size in each group if equal.sample = TRUE
`n2`	sample size in group 2
`power`	power of test (1 minus Type II error probability)
`sig.level`	significance level (Type I error probability)
`mu1`	arithmetic mean of group 1
`mu2`	arithmetic mean of group 2
`gmu1`	geometric mean of group 1
`gmu2`	geometric mean of group 2
`trials`	number of trials in simulation
`M`	number of simulations used in CAT method, see Chang (2011)
`equal.sample`	equal sample sizes for two groups, see details
`equal.shape`	assume the shape parameters are equal for two groups, see details

Details

Exactly one of the parameters n1, n2, and power must be passed as NULL, and that parameter is determined from the others. Notice that sig.level has non-NULL defaults, so NULL must be explicitly passed if you want to compute it.

If equal.sample = TRUE is used, N in output will denote the number in each group.

The equal shape parameter assumption will be tested automatically; otherwise it could be set manually with equal.shape.

Value

Object of class "power.htest", a list of the arguments (including the computed one) augmented with method element.

References

Chang et al. (2011). Testing the equality of several gamma means: a parametric bootstrap method with applications. *Computational Statistics*, **26**:55-76.

Examples

```
# Calculate power, equal sizes
power_Gamma(n1 = 50, mu1 = 1, mu2 = 1.5, gmu1 = 0.6, gmu2 = 0.6, M = 100)
```

---

power_Geometric *Power Calculation for Comparing Two Geometric Rates.*

Description

Compute sample size or power for comparing two geometric rates.

Usage

```
power_Geometric(n1 = NULL, n2 = NULL, power = NULL, sig.level = 0.05,
mu1 = NULL, mu2 = NULL, duration = 1, equal.sample = TRUE,
alternative = c("two.sided", "one.sided"), approach = 3)
```

Arguments

n1	sample size in group 1, or sample size in each group if equal.sample = TRUE
n2	sample size in group 2
power	power of test (1 minus Type II error probability)
sig.level	significance level (Type I error probability)
mu1	expected rate of events per time unit for group 1
mu2	expected rate of events per time unit for group 2

`duration`	(average) treatment duration
`equal.sample`	equal sample sizes for two groups, see details
`alternative`	one- or two-sided test
`approach`	1, 2, or 3; see Zhu and Lakkis (2014).

Details

Exactly one of the parameters `n1`, `n2`, and `power` must be passed as NULL, and that parameter is determined from the others.

If `equal.sample = TRUE` is used, N in output will denote the number in each group.

Since the geometric distribution is a special case of negative binomial distribution, we used the algorithm for negative binomial distribution with setting theta = 1. See power.nb.test for more details.

Value

Object of class "power.htest", a list of the arguments (including the computed one) augmented with note and method elements.

Examples

```
# calculate power, equal sizes
power_Geometric(n1 = 100, mu1 = 0.3, mu2 = 0.6)
# calculate power, unequal sizes
power_Geometric(n1 = 180, n2 = 140, mu1 = 0.3, mu2 = 0.5)
# calculate n
power_Geometric( mu1 = 0.3, mu2 = 0.4, power = 0.8)
```

---

power_NegativeBinomial *Power Calculation for Comparing Two Negative Binomial Rates*

Description

Compute sample size or power for comparing two negative binomial rates.

Usage

```
power_NegativeBinomial(n1 = NULL, n2 = NULL, power = NULL, sig.level = 0.05,
mu1 = NULL, mu2 = NULL, duration = 1, theta = NULL, equal.sample = TRUE,
alternative = c("two.sided", "one.sided"), approach = 3)
```

Arguments

`n1`	sample size in group 1, or sample size in each group if `equal.sample = TRUE`
`n2`	sample size in group 2
`power`	power of test (1 minus Type II error probability)
`sig.level`	significance level (Type I error probability)
`mu1`	expected rate of events per time unit for group 1
`mu2`	expected rate of events per time unit for group 2
`duration`	(average) treatment duration
`theta`	theta parameter of negative binomial distribution; see rnegbin
`equal.sample`	equal sample sizes for two groups, see details
`alternative`	one- or two-sided test
`approach`	1, 2, or 3; see Zhu and Lakkis (2014).

Details

Exactly one of the parameters `n1`, `n2`, and `power` must be passed as NULL, and that parameter is determined from the others.

If `equal.sample = TRUE` is used, N in output will denote the number in each group.

The computations are based on the formulas given in Zhu and Lakkis (2014). See power.nb.test for more details.

Value

Object of class "power.htest", a list of the arguments (including the computed one) augmented with note and method elements.

References

H. Zhu and H. Lakkis (2014). Sample size calculation for comparing two negative binomial rates. *Statistics in Medicine*, **33**:376-387.

Examples

```
# calculate power, equal sizes
power_NegativeBinomial(n1 = 20, mu1 = 1, mu2 = 2, theta = 0.8)
# calculate power, unequal sizes
power_NegativeBinomial(n1 = 80, n2 = 40, mu1 = 1, mu2 = 2, theta = 0.8)
# calculate n
power_NegativeBinomial( mu1 = 1, mu2 = 2, theta = 0.8, power = 0.8)
```

`power_Normal` *Power Calculations for One and Two Sample T-tests*

Description

Compute power of t test, or determine parameters to obtain target power.

Usage

```
power_Normal(n1 = NULL, n2 = NULL, power = NULL, sig.level = 0.05,
delta = NULL, sd1 = 1, sd2 = 1, equal.sample = TRUE,
alternative = c("two.sided", "one.sided"),
type = c("two.sample", "one.sample", "paired"),
df.method = c("welch", "classical"), strict = FALSE)
```

Arguments

`n1`	sample size in group 1, or sample size in each group if equal.sample = TRUE
`n2`	sample size in group 2
`power`	power of test (1 minus Type II error probability)
`sig.level`	significance level (Type I error probability)
`delta`	true difference in means
`sd1`	standard deviation for group 1
`sd2`	standard deviation for group 2
`equal.sample`	equal sample sizes for two groups, see details
`alternative`	one- or two-sided test
`type`	Type of t test
`df.method`	Method for calculating the degrees of default. Possibilities are welch (the default) or classical.
`strict`	Use strict interpretation in two-sided case

Details

Exactly one of the parameters `n1`, `n2`, `delta`, `sd1`, `sd2`, `power`, and `sig.level` must be passed as NULL, and that parameter is determined from the others. Notice that `sd1`, `sd2`, `sig.level` have non-NULL defaults, so NULL must be explicitly expressed if you want to compute them.

If `equal.sample = TRUE` is used, N in output will denote the number in each group.

Value

Object of class "power.htest", a list of the arguments (including the computed one) augmented with note and method elements.

Note

'uniroot' is used to solve power equation for unknowns, so you may see errors from it, notably about inability to bracket the root when invalid arguments are given.

Examples

```
# Calculate power, equal sizes
power_Normal(n1 = 150, delta = 5, sd1 = 20, sd2 = 10)
# Calculate power, unequal sizes
power_Normal(n1 = 150, delta = 5, n2 = 120, sd1 = 10)
# Calculate n1, equal sizes
power_Normal(delta = 5,  power = 0.9, sd1 = 10, sd2 = 12)
```

`power_Poisson` *Power Calculations for Test of Two Poisson Ratios*

Description

Compute the power for a test of two sample means with Poisson distributions, or determine parameters to obtain a target power.

Usage

```
power_Poisson(n1 = NULL, n2 = NULL, power = NULL, sig.level = 0.05,
lambda1 = NULL, lambda2 = NULL, t1 = 1, t2 = 1, RR0 = 1,
equal.sample = TRUE, alternative = c("two.sided", "one.sided"))
```

Arguments

`n1`	sample size in group 1, or sample size in each group if `equal.sample = TRUE`
`n2`	sample size in group 2
`power`	power of test (1 minus Type II error probability)
`sig.level`	significance level (Type I error probability)
`lambda1`	Poisson rate for group 1
`lambda2`	Poisson rate for group 2
`t1`	observed time period for group 1
`t2`	observed time period for group 2
`RR0`	the ratio of lambda2 and lambda1 under null hypothesis
`equal.sample`	equal sample sizes for two groups, see details
`alternative`	one- or two-sided test

Details

Exactly one of the parameters n1, n2, lambda1, lambda2, power, and sig.level must be passed as NULL, and that parameter is determined from the others. Notice that sig.level has non-NULL defaults, so NULL must be explicitly passed if you want to compute them.

If equal.sample = TRUE is used, n2 would be ignored and N in output denotes the number in each group.

Value

Object of class "power.htest", a list of the arguments (including the computed one) augmented with method element.

Note

'uniroot' is used to solve power equation for unknowns, so you may see errors from it, notably about inability to bracket the root when invalid arguments are given.

References

Gu et al. (2008). Testing the ratio of two poisson rates. *Biometrical Journal: Journal of Mathematical Methods in Biosciences*. **50**:283-298.

Examples

```
# Calculate power, equal sizes
power_Poisson(lambda1 = 0.0005, lambda2 = 0.003, n1 = 2000, t1 = 2, t2 = 2)
# Calculate sample size, equal sizes
power_Poisson(lambda1 = 0.0005, lambda2 = 0.003, power = 0.8, t1 = 2, t2 = 2)
# Calculate sample size for group 2, unequal sizes
power_Poisson(n1 = 2000, lambda1 = 0.0005, lambda2 = 0.003, power = 0.8,
t1 = 2, t2 = 2, equal.sample = FALSE)
```

Index

power.nb.test, *6*, *7*
power_Beta, 2
power_Binomial, 3
power_Gamma, 4
power_Geometric, 5
power_NegativeBinomial, 6
power_Normal, 8
power_Poisson, 9

rnegbin, *7*