Package 'MKpower'

October 12, 2022

Version 0.5

Date 2020-11-27

Title Power Analysis and Sample Size Calculation

Author Matthias Kohl [aut, cre] (<https://orcid.org/0000-0001-9514-8910>)

Maintainer Matthias Kohl <Matthias.Kohl@stamats.de>

Depends R(>= 3.5.0)

Imports stats, matrixTests, ggplot2, MKdescr, MKinfer(>= 0.4), qqplotr, coin

Suggests knitr, rmarkdown

VignetteBuilder knitr

Description Power analysis and sample size calculation for Welch and Hsu (Hed-derich and Sachs (2018), ISBN:978-3-662-56657-2) t-tests including Monte-Carlo simulations of empirical power and type-I-error. Power and sample size calculation for Wilcoxon rank sum and signed rank tests via Monte-Carlo simulations. Power and sample size required for the evaluation of a diagnostic test(-system) (Flahault et al. (2005), <doi:10.1016/j.jclinepi.2004.12.009>; Dobbin and Simon (2007), <doi:10.1093/biostatistics/kxj036>) as well as for a single proportion (Fleiss et al. (2003), ISBN:978-0-471-52629-2; Piegorsch (2004), <doi:10.1016/j.csda.2003.10.002>; Thulin (2014), <doi:10.1214/14-ejs909>) and comparing two negative binomial rates (Zhu and Lakkis (2014), <doi:10.1002/sim.5947>).

License LGPL-3

URL http://www.stamats.de/

NeedsCompilation no

Repository CRAN

Date/Publication 2020-11-28 14:50:10 UTC

R topics documented:

`MKpower-package` *Power Analysis and Sample Size Calculation.*

Description

Power analysis and sample size calculation for Welch and Hsu (Hedderich and Sachs (2018), ISBN:978-3-662-56657-2) t-tests including Monte-Carlo simulations of empirical power and type-I-error. Power and sample size calculation for Wilcoxon rank sum and signed rank tests via Monte-Carlo simulations. Power and sample size required for the evaluation of a diagnostic test(-system) (Flahault et al. (2005), <doi:10.1016/j.jclinepi.2004.12.009>; Dobbin and Simon (2007), <doi:10.1093/biostatistics/kxj036>) as well as for a single proportion (Fleiss et al. (2003), ISBN:978-0-471-52629-2; Piegorsch (2004), <doi:10.1016/j.csda.2003.10.002>; Thulin (2014), <doi:10.1214/14-ejs909>) and comparing two negative binomial rates (Zhu and Lakkis (2014), <doi:10.1002/sim.5947>).

Details

Package:	MKpower
Type:	Package
Version:	0.5
Date:	2020-11-27
Depends:	R(>= 3.5.0)
Imports:	stats, matrixTests, ggplot2, MKdescr, MKinfer(>= 0.4), qqplotr, coin
Suggests:	knitr, rmarkdown
License:	LGPL-3
URL:	http://www.stamats.de/

```
library(MKpower)
```

Author(s)

Matthias Kohl http://www.stamats.de

Maintainer: Matthias Kohl <matthias.kohl@stamats.de>

`hist` *Histograms*

Description

Produce histograms for simulations of power and type-I-error of tests.

Usage

```
## S3 method for class 'sim.power.ttest'
hist(x, color.hline = "orange", ...)

## S3 method for class 'sim.power.wtest'
hist(x, color.hline = "orange", ...)
```

Arguments

`x`	object of class `sim.power.ttest`.
`color.hline`	color of horizontal line indicating uniform distribution of p values.
`...`	further arguments that may be passed through).

Details

The plot generates a `ggplot2` object that is shown.

Missing values are handled by the `ggplot2` functions.

Value

Object of class gg and ggplot.

Author(s)

Matthias Kohl <Matthias.Kohl@stamats.de>

See Also

`hist`

Examples

```
res1 <- sim.power.t.test(nx = 5, rx = rnorm, rx.H0 = rnorm,
                         ny = 10, ry = function(x) rnorm(x, mean = 3, sd = 3),
                         ry.H0 = function(x) rnorm(x, sd = 3))
hist(res1)
res2 <- sim.power.wilcox.test(nx = 6, rx = rnorm, rx.H0 = rnorm,
                       ny = 6, ry = function(x) rnorm(x, mean = 2),
                       ry.H0 = rnorm)
hist(res2)
```

---

power.diagnostic.test *Power calculations for a diagnostic test*

Description

Compute sample size, power, delta, or significance level of a diagnostic test for an expected sensitivy or specificity.

Usage

```
power.diagnostic.test(sens = NULL, spec = NULL,
                      n = NULL, delta = NULL, sig.level = 0.05,
                      power = NULL, prev = NULL,
                      method = c("exact", "asymptotic"),
                      NMAX = 1e4)
```

Arguments

sens	Expected sensitivity; either sens or spec has to be specified.
spec	Expected specificity; either sens or spec has to be specified.
n	Number of cases if sens and number of controls if spec is given.
delta	sens-delta resp. spec-delta is used as lower confidence limit
sig.level	Significance level (Type I error probability)
power	Power of test (1 minus Type II error probability)
prev	Expected prevalence, if NULL prevalence is ignored which means prev = 0.5 is assumed.
method	exact or asymptotic formula; default "exact".
NMAX	Maximum sample size considered in case method = "exact".

Details

Either sens or spec has to be specified which leads to computations for either cases or controls.

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

The computations are based on the formulas given in the Appendix of Flahault et al. (2005). Please be careful, in Equation (A1) the numerator should be squared, in equation (A2) and (A3) the second exponent should be n-i and not i.

As noted in Chu and Cole (2007) power is not a monotonically increasing function in n but rather saw toothed (see also Chernick and Liu (2002)). Hence, in our calculations we use the more conservative approach II); i.e., the minimum sample size n such that the actual power is larger or equal power andsuch that for any sample size larger than n it also holds that the actual power is larger or equal power.

Value

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

Author(s)

Matthias Kohl <Matthias.Kohl@stamats.de>

References

A. Flahault, M. Cadilhac, and G. Thomas (2005). Sample size calculation should be performed for design accuracy in diagnostic test studies. *Journal of Clinical Epidemiology*, **58**(8):859-862.

H. Chu and S.R. Cole (2007). Sample size calculation using exact methods in diagnostic test studies. *Journal of Clinical Epidemiology*, **60**(11):1201-1202.

M.R. Chernick amd C.Y. Liu (2002). The saw-toothed behavior of power versus sample size and software solutions: single binomial proportion using exact methods. *Am Stat*, **56**:149-155.

See Also

uniroot

Examples

```
## see n2 on page 1202 of Chu and Cole (2007)
power.diagnostic.test(sens = 0.99, delta = 0.14, power = 0.95) # 40
power.diagnostic.test(sens = 0.99, delta = 0.13, power = 0.95) # 43
power.diagnostic.test(sens = 0.99, delta = 0.12, power = 0.95) # 47
```

```
power.diagnostic.test(sens = 0.98, delta = 0.13, power = 0.95) # 50
power.diagnostic.test(sens = 0.98, delta = 0.11, power = 0.95) # 58

## see page 1201 of Chu and Cole (2007)
power.diagnostic.test(sens = 0.95, delta = 0.1, n = 93) ## 0.957
power.diagnostic.test(sens = 0.95, delta = 0.1, n = 93, power = 0.95,
                      sig.level = NULL) ## 0.0496
power.diagnostic.test(sens = 0.95, delta = 0.1, n = 102) ## 0.968
power.diagnostic.test(sens = 0.95, delta = 0.1, n = 102, power = 0.95,
                      sig.level = NULL) ## 0.0471
## yields 102 not 93!
power.diagnostic.test(sens = 0.95, delta = 0.1, power = 0.95)
```

`power.hsu.t.test` *Power calculations for two sample Hsu t test*

Description

Compute the power of the two-sample Hsu t test, or determine parameters to obtain a target power; see Section 7.4.4 in Hedderich and Sachs (2016),

Usage

```
power.hsu.t.test(n = NULL, delta = NULL, sd1 = 1, sd2 = 1, sig.level = 0.05,
                 power = NULL, alternative = c("two.sided", "one.sided"),
                 strict = FALSE, tol = .Machine$double.eps^0.25)
```

Arguments

`n`	number of observations (per group)
`delta`	(expected) true difference in means
`sd1`	(expected) standard deviation of group 1
`sd2`	(expected) standard deviation of group 2
`sig.level`	significance level (Type I error probability)
`power`	power of test (1 minus Type II error probability)
`alternative`	one- or two-sided test. Can be abbreviated.
`strict`	use strict interpretation in two-sided case
`tol`	numerical tolerance used in root finding, the default providing (at least) four significant digits.

Details

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

If `strict = TRUE` is used, the power will include the probability of rejection in the opposite direction of the true effect, in the two-sided case. Without this the power will be half the significance level if the true difference is zero.

Value

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

Note

The function and its documentation was adapted from power.t.test implemented by Peter Dalgaard and based on previous work by Claus Ekstroem.

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

Author(s)

Matthias Kohl <Matthias.Kohl@stamats.de>

References

J. Hedderich, L. Sachs. *Angewandte Statistik: Methodensammlung mit R*. Springer 2016.

See Also

power.welch.t.test, power.t.test, t.test, uniroot

Examples

```
## more conservative than classical or Welch t-test
power.hsu.t.test(n = 20, delta = 1)
power.hsu.t.test(power = .90, delta = 1)
power.hsu.t.test(power = .90, delta = 1, alternative = "one.sided")

## sd1 = 0.5, sd2 = 1
power.welch.t.test(delta = 0.5, sd1 = 0.5, sd2 = 1, power = 0.9)
power.hsu.t.test(delta = 0.5, sd1 = 0.5, sd2 = 1, power = 0.9)


if(require(MKinfer)){
## empirical check
M <- 10000
ps <- numeric(M)
for(i in seq_len(M)){
  x <- rnorm(55, mean = 0, sd = 0.5)
  y <- rnorm(55, mean = 0.5, sd = 1.0)
  ps[i] <- hsu.t.test(x, y)$p.value
}
## empirical power
sum(ps < 0.05)/M
}
```

power.nb.test *Power calculation for comparing two negative binomial rates*

Description

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

Usage

```
power.nb.test(n = NULL, mu0, mu1, RR, duration = 1, theta, ssize.ratio = 1,
         sig.level = 0.05, power = NULL, alternative = c("two.sided", "one.sided"),
              approach = 3)
```

Arguments

n	Sample size for group 0 (control group).
mu0	expected rate of events per time unit for group 0
mu1	expected rate of events per time unit for group 1
RR	ratio of expected event rates: mu1/mu0
duration	(average) treatment duration
theta	theta parameter of negative binomial distribution; see rnegbin
ssize.ratio	ratio of sample sizes: n/n1 where n1 is sample size of group 1
sig.level	Significance level (Type I error probability)
power	Power of test (1 minus Type II error probability)
alternative	one- or two-sided test
approach	1, 2, or 3; see Zhu and Lakkis (2014).

Details

Exactly one of the parameters n and power must be passed as NULL, and that parameter is determined from the other.

The computations are based on the formulas given in Zhu and Lakkis (2014). Please be careful, as we are using a slightly different parametrization (theta = 1/k).

Zhu and Lakkis (2014) based on their simulation studies recommend to use their approach 2 or 3.

Value

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

Author(s)

Matthias Kohl <Matthias.Kohl@stamats.de>

References

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

See Also

rnegbin, glm.nb

Examples

```
## examples from Table I in Zhu and Lakkis (2014)
## theta = 1/k, RR = rr, mu0 = r0, duration = mu_t
power.nb.test(mu0 = 0.8, RR = 0.85, theta = 1/0.4, duration = 0.75, power = 0.8, approach = 1)
power.nb.test(mu0 = 0.8, RR = 0.85, theta = 1/0.4, duration = 0.75, power = 0.8, approach = 2)
power.nb.test(mu0 = 0.8, RR = 0.85, theta = 1/0.4, duration = 0.75, power = 0.8, approach = 3)

power.nb.test(mu0 = 1.4, RR = 1.15, theta = 1/1.5, duration = 0.75, power = 0.8, approach = 1)
power.nb.test(mu0 = 1.4, RR = 1.15, theta = 1/1.5, duration = 0.75, power = 0.8, approach = 2)
power.nb.test(mu0 = 1.4, RR = 1.15, theta = 1/1.5, duration = 0.75, power = 0.8, approach = 3)


## examples from Table II in Zhu and Lakkis (2014) - seem to be total sample sizes
## can reproduce the results with mu_t = 1.0 (not 0.7!)
power.nb.test(mu0 = 2.0, RR = 0.5, theta = 1, duration = 1.0, ssize.ratio = 1,
              power = 0.8, approach = 1)
power.nb.test(mu0 = 2.0, RR = 0.5, theta = 1, duration = 1.0, ssize.ratio = 1,
              power = 0.8, approach = 2)
power.nb.test(mu0 = 2.0, RR = 0.5, theta = 1, duration = 1.0, ssize.ratio = 1,
              power = 0.8, approach = 3)

power.nb.test(mu0 = 10.0, RR = 1.5, theta = 1/5, duration = 1.0, ssize.ratio = 3/2,
              power = 0.8, approach = 1)
power.nb.test(mu0 = 10.0, RR = 1.5, theta = 1/5, duration = 1.0, ssize.ratio = 3/2,
              power = 0.8, approach = 2)
power.nb.test(mu0 = 10.0, RR = 1.5, theta = 1/5, duration = 1.0, ssize.ratio = 3/2,
              power = 0.8, approach = 3)


## examples from Table III in Zhu and Lakkis (2014)
power.nb.test(mu0 = 5.0, RR = 2.0, theta = 1/0.5, duration = 1, power = 0.8, approach = 1)
power.nb.test(mu0 = 5.0, RR = 2.0, theta = 1/0.5, duration = 1, power = 0.8, approach = 2)
power.nb.test(mu0 = 5.0, RR = 2.0, theta = 1/0.5, duration = 1, power = 0.8, approach = 3)


## examples from Table IV in Zhu and Lakkis (2014)
power.nb.test(mu0 = 5.9/3, RR = 0.4, theta = 0.49, duration = 3, power = 0.9, approach = 1)
power.nb.test(mu0 = 5.9/3, RR = 0.4, theta = 0.49, duration = 3, power = 0.9, approach = 2)
power.nb.test(mu0 = 5.9/3, RR = 0.4, theta = 0.49, duration = 3, power = 0.9, approach = 3)

power.nb.test(mu0 = 13/6, RR = 0.2, theta = 0.52, duration = 6, power = 0.9, approach = 1)
power.nb.test(mu0 = 13/6, RR = 0.2, theta = 0.52, duration = 6, power = 0.9, approach = 2)
power.nb.test(mu0 = 13/6, RR = 0.2, theta = 0.52, duration = 6, power = 0.9, approach = 3)
```

```
## see Section 5 of Zhu and Lakkis (2014)
power.nb.test(mu0 = 0.66, RR = 0.8, theta = 1/0.8, duration = 0.9, power = 0.9)
```

`power.prop1.test` *Power Calculations for One-Sample Test for Proportions*

Description

Compute the power of the one-sample test for proportions, or determine parameters to obtain a target power.

Usage

```
power.prop1.test(n = NULL, p1 = NULL, p0 = 0.5, sig.level = 0.05,
                                power = NULL,
                                alternative = c("two.sided", "less", "greater"),
                                cont.corr = TRUE, tol = .Machine$double.eps^0.25)
```

Arguments

`n`	number of observations (per group)
`p1`	expected probability
`p0`	probability under the null hypothesis
`sig.level`	significance level (Type I error probability)
`power`	power of test (1 minus Type II error probability)
`alternative`	one- or two-sided test. Can be abbreviated.
`cont.corr`	use continuity correction
`tol`	numerical tolerance used in root finding, the default providing (at least) four significant digits.

Details

Exactly one of the parameters `n`, `p1`, `power`, and `sig.level` must be passed as NULL, and that parameter is determined from the others. Notice that `sig.level` has a non-NULL default so `NULL` must be explicitly passed if you want it computed.

The computation is based on the asymptotic formulas provided in Section 2.5.1 of Fleiss et al. (2003). If `cont.corr = TRUE` a continuity correction is applied, which may lead to better approximations of the finite-sample values.

Value

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

Note

The documentation was adapted from power.prop.test.

Author(s)

Matthias Kohl <Matthias.Kohl@stamats.de>

References

J.L. Fleiss, B. Levin and M.C. Paik (2003). *Statistical Methods for Rates and Proportions*. Wiley Series in Probability and Statistics.

See Also

power.prop.test, prop.test

Examples

```
power.prop1.test(p1 = 0.4, power = 0.8)
power.prop1.test(p1 = 0.4, power = 0.8, cont.corr = FALSE)
power.prop1.test(p1 = 0.6, power = 0.8)
power.prop1.test(n = 204, power = 0.8)
power.prop1.test(n = 204, p1 = 0.4, power = 0.8, sig.level = NULL)
power.prop1.test(n = 194, p1 = 0.4, power = 0.8, sig.level = NULL,
                 cont.corr = FALSE)

power.prop1.test(p1 = 0.1, p0 = 0.3, power = 0.8, alternative = "less")
power.prop1.test(p1 = 0.1, p0 = 0.3, power = 0.8, alternative = "less",
                 cont.corr = FALSE)
power.prop1.test(n = 31, p0 = 0.3, power = 0.8, alternative = "less")
power.prop1.test(n = 31, p1 = 0.1, p0 = 0.3, power = 0.8, sig.level = NULL,
                 alternative = "less")


power.prop1.test(p1 = 0.5, p0 = 0.3, power = 0.8, alternative = "greater")
power.prop1.test(p1 = 0.5, p0 = 0.3, power = 0.8, alternative = "greater",
                 cont.corr = FALSE)
power.prop1.test(n = 40, p0 = 0.3, power = 0.8, alternative = "greater")
power.prop1.test(n = 40, p1 = 0.5, p0 = 0.3, power = 0.8, sig.level = NULL,
                 alternative = "greater")
```

---

power.welch.t.test *Power calculations for two sample Welch t test*

Description

Compute the power of the two-sample Welch t test, or determine parameters to obtain a target power.

Usage

```
power.welch.t.test(n = NULL, delta = NULL, sd1 = 1, sd2 = 1, sig.level = 0.05,
                   power = NULL, alternative = c("two.sided", "one.sided"),
                   strict = FALSE, tol = .Machine$double.eps^0.25)
```

Arguments

n	number of observations (per group)
delta	(expected) true difference in means
sd1	(expected) standard deviation of group 1
sd2	(expected) standard deviation of group 2
sig.level	significance level (Type I error probability)
power	power of test (1 minus Type II error probability)
alternative	one- or two-sided test. Can be abbreviated.
strict	use strict interpretation in two-sided case
tol	numerical tolerance used in root finding, the default providing (at least) four significant digits.

Details

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

If strict = TRUE is used, the power will include the probability of rejection in the opposite direction of the true effect, in the two-sided case. Without this the power will be half the significance level if the true difference is zero.

Value

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

Note

The function and its documentation was adapted from power.t.test implemented by Peter Dalgaard and based on previous work by Claus Ekstroem.

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

Author(s)

Matthias Kohl <Matthias.Kohl@stamats.de>

References

S.L. Jan and G. Shieh (2011). Optimal sample sizes for Welch's test under various allocation and cost considerations. *Behav Res Methods*, 43, 4:1014-22.

See Also

power.t.test, t.test, uniroot

Examples

```
## identical results as power.t.test, since sd = sd1 = sd2 = 1
power.welch.t.test(n = 20, delta = 1)
power.welch.t.test(power = .90, delta = 1)
power.welch.t.test(power = .90, delta = 1, alternative = "one.sided")

 ## sd1 = 0.5, sd2 = 1
power.welch.t.test(delta = 2, sd1 = 0.5, sd2 = 1, power = 0.9)


## empirical check
M <- 10000
pvals.welch <- numeric(M)
for(i in seq_len(M)){
  x <- rnorm(5, mean = 0, sd = 0.5)
  y <- rnorm(5, mean = 2, sd = 1.0)
  pvals.welch[i] <- t.test(x, y)$p.value
}
## empirical power
sum(pvals.welch < 0.05)/M
```

---

qqunif *qq - Plots for Uniform Distribution*

Description

Produce qq-plot(s) of the given effect size and p values assuming a uniform distribution.

Usage

```
qqunif(x, ...)

## Default S3 method:
qqunif(x, min = 0, max = 1, ...)

## S3 method for class 'sim.power.ttest'
qqunif(x, color.line = "orange", shape = 19, size = 1,
                          alpha = 1, ...)

## S3 method for class 'sim.power.wtest'
qqunif(x, color.line = "orange", shape = 19, size = 1,
                          alpha = 1, ...)
```

Arguments

x	numeric vector or data (object).
min	single numeric, lower limit of the distribution.
max	single numeric, upper limit of the distribution.
color.line	color of the line indicating the uniform distribution.
shape	point shape.
size	point size.
alpha	bleding factor (default: no blending.
...	further arguments that may be passed through).

Details

The plot generates a ggplot2 object that is shown.

Missing values are handled by the ggplot2 functions.

Value

Object of class gg and ggplot.

Author(s)

Matthias Kohl <Matthias.Kohl@stamats.de>

Examples

```
## default
qqunif(runif(100))

## visualization of empirical power and type-I-error
res1 <- sim.power.t.test(nx = 5, rx = rnorm, rx.H0 = rnorm,
                         ny = 10, ry = function(x) rnorm(x, mean = 3, sd = 3),
                         ry.H0 = function(x) rnorm(x, sd = 3))
qqunif(res1, alpha = 0.1)

res2 <- sim.power.wilcox.test(nx = 6, rx = rnorm, rx.H0 = rnorm,
                       ny = 6, ry = function(x) rnorm(x, mean = 2),
                       ry.H0 = rnorm)
qqunif(res2)
```

`sim.power.t.test` *Monte Carlo Simulations for Empirical Power of Two-sample t-Tests*

Description

Simulate the empirical power and type-I-error of two-sample t-tests; i.e., classical (equal variances), Welch and Hsu t-tests.

Usage

```
sim.power.t.test(nx, rx, rx.H0 = NULL, ny, ry, ry.H0 = NULL,
                 sig.level = 0.05, mu = 0,
                 alternative = c("two.sided", "less", "greater"),
                 iter = 10000)
```

Arguments

`nx`	single numeric, sample size of first group.
`rx`	function to simulate the values of first group (assuming H1).
`rx.H0`	`NULL` or function to simulate the values of first group (assuming H0).
`ny`	single numeric, sample size of second group.
`ry`	function to simulate the values of second group (assuming H1).
`ry.H0`	`NULL` or function to simulate the values of second group (assuming H0).
`sig.level`	significance level (type I error probability)
`mu`	true value of the location shift for the null hypothesis.
`alternative`	one- or two-sided test. Can be abbreviated.
`iter`	single integer, number of interations of the simulations.

Details

Functions `rx` and `ry` are used to simulate the data under the alternative hypothesis H1. If specified, functions `rx.H0` and `ry.H0` simulte the data unter the null hypothesis H0.

For fast computations functions from package `matrixTests` are used.

Value

Object of class `"sim.power.ttest"` with the results of the three t-tests in the list elements `Classical`, `Welch` and `Hsu`. In addition, the simulation setup is saved in element `SetUp`.

Author(s)

Matthias Kohl <Matthias.Kohl@stamats.de>

References

J. Hedderich, L. Sachs. *Angewandte Statistik: Methodensammlung mit R*. Springer 2018.

Hsu, P. (1938). Contribution to the theory of "student's" t-test as applied to the problem of two samples. *Statistical Research Memoirs* **2**: 1-24.

Student (1908). The Probable Error of a Mean. *Biometrika*, **6**(1): 1-25.

Welch, B. L. (1947). The generalization of "Student's" problem when several different population variances are involved. *Biometrika*, **34** (1-2): 28-35.

See Also

`t.test`, `hsu.t.test`, `ttest`

Examples

```
## Equal variance, small sample size
power.t.test(n = 5, delta = 2)
power.welch.t.test(n = 5, delta = 2)
power.hsu.t.test(n = 5, delta = 2)
sim.power.t.test(nx = 5, rx = rnorm, rx.H0 = rnorm,
                 ny = 5, ry = function(x) rnorm(x, mean = 2), ry.H0 = rnorm)

## Equal variance, moderate sample size
power.t.test(n = 25, delta = 0.8)
power.welch.t.test(n = 25, delta = 0.8)
power.hsu.t.test(n = 25, delta = 0.8)
sim.power.t.test(nx = 25, rx = rnorm, rx.H0 = rnorm,
                 ny = 25, ry = function(x) rnorm(x, mean = 0.8), ry.H0 = rnorm)

## Equal variance, high sample size
power.t.test(n = 100, delta = 0.4)
power.welch.t.test(n = 100, delta = 0.4)
power.hsu.t.test(n = 100, delta = 0.4)
sim.power.t.test(nx = 100, rx = rnorm, rx.H0 = rnorm,
                 ny = 100, ry = function(x) rnorm(x, mean = 0.4), ry.H0 = rnorm)

## Unequal variance, small sample size
power.welch.t.test(n = 5, delta = 5, sd1 = 1, sd2 = 3)
power.hsu.t.test(n = 5, delta = 5, sd1 = 1, sd2 = 3)
sim.power.t.test(nx = 5, rx = rnorm, rx.H0 = rnorm,
                 ny = 5, ry = function(x) rnorm(x, mean = 5, sd = 3),
                 ry.H0 = function(x) rnorm(x, sd = 3))

## Unequal variance, moderate sample size
power.welch.t.test(n = 25, delta = 1.8, sd1 = 1, sd2 = 3)
power.hsu.t.test(n = 25, delta = 1.8, sd1 = 1, sd2 = 3)
sim.power.t.test(nx = 25, rx = rnorm, rx.H0 = rnorm,
                 ny = 25, ry = function(x) rnorm(x, mean = 1.8, sd = 3),
                 ry.H0 = function(x) rnorm(x, sd = 3))

## Unequal variance, high sample size
```

```
power.welch.t.test(n = 100, delta = 0.9, sd1 = 1, sd2 = 3)
power.hsu.t.test(n = 100, delta = 0.9, sd1 = 1, sd2 = 3)
sim.power.t.test(nx = 100, rx = rnorm, rx.H0 = rnorm,
                 ny = 100, ry = function(x) rnorm(x, mean = 0.9, sd = 3),
                 ry.H0 = function(x) rnorm(x, sd = 3))

## Unequal variance, unequal sample sizes
## small sample sizes
sim.power.t.test(nx = 10, rx = rnorm, rx.H0 = rnorm,
                 ny = 5, ry = function(x) rnorm(x, mean = 5, sd = 3),
                 ry.H0 = function(x) rnorm(x, sd = 3))
sim.power.t.test(nx = 5, rx = rnorm, rx.H0 = rnorm,
                 ny = 10, ry = function(x) rnorm(x, mean = 3, sd = 3),
                 ry.H0 = function(x) rnorm(x, sd = 3))

## Unequal variance, unequal sample sizes
## moderate sample sizes
sim.power.t.test(nx = 25, rx = rnorm, rx.H0 = rnorm,
                 ny = 50, ry = function(x) rnorm(x, mean = 1.5, sd = 3),
                 ry.H0 = function(x) rnorm(x, sd = 3))

## Unequal variance, unequal sample sizes
## high sample sizes
sim.power.t.test(nx = 100, rx = rnorm, rx.H0 = rnorm,
                 ny = 200, ry = function(x) rnorm(x, mean = 0.6, sd = 3),
                 ry.H0 = function(x) rnorm(x, sd = 3))
```

`sim.power.wilcox.test` *Monte Carlo Simulations for Empirical Power of Wilcoxon-Mann-Whitney Tests*

Description

Simulate the empirical power and type-I-error of Wilcoxon-Mann-Whitney tests.

Usage

```
sim.power.wilcox.test(nx, rx, rx.H0 = NULL, ny, ry, ry.H0 = NULL,
                      alternative = c("two.sided", "less", "greater"),
                      sig.level = 0.05, conf.int = FALSE, approximate = FALSE,
                      ties = FALSE, iter = 10000, nresample = 10000,
                      parallel = "no", ncpus = 1L, cl = NULL)
```

Arguments

`nx`	single numeric, sample size of first group.
`rx`	function to simulate the values of first group (assuming H1).
`rx.H0`	`NULL` or function to simulate the values of first group (assuming H0).

ny	single numeric, sample size of second group.
ry	function to simulate the values of second group (assuming H1).
ry.H0	NULL or function to simulate the values of second group (assuming H0).
alternative	one- or two-sided test. Can be abbreviated.
sig.level	significance level (type I error probability)
conf.int	logical, shall confidence intervals be computed. Strongly increases computation time!
approximate	logical, shall an approximate test be computed; see LocationTests. Increases computation time!
ties	logical, indicating whether ties may occur. Increases computation time!
iter	single positive integer, number of interations of the simulations.
nresample	single positive integer, the number of Monte Carlo replicates used for the computation of the approximative reference distribution; see NullDistribution.
parallel	a character, the type of parallel operation: either "no" (default), "multicore" or "snow"; see NullDistribution.
ncpus	a single integer, the number of processes to be used in parallel operation. Defaults to 1L; see NullDistribution.
cl	an object inheriting from class "cluster", specifying an optional parallel or snow cluster if parallel = "snow". Defaults to NULL; see NullDistribution.

Details

Functions rx and ry are used to simulate the data under the alternative hypothesis H1. If specified, functions rx.H0 and ry.H0 simulte the data unter the null hypothesis H0.

For fast computations functions from package matrixTests and package coin are used.

Value

Object of class "sim.power.wtest" with the results of the Wilcoxon-Mann-Whitney tests. A list elements Exact, Asymptotic and Approximate. In addition, the simulation setup is saved in element SetUp.

Author(s)

Matthias Kohl <Matthias.Kohl@stamats.de>

References

Mann, H and Withney, D (1947). On a test of whether one of two random variables is stochastically larger than the other. *Annals of mathematical Statistics*, **18**, 50-60.

Wilcoxon, F (1945). Individual Comparisons by Ranking Methods. *Biometrics Bulletin*, **1**, 80-83.

See Also

wilcox.test, LocationTests, wilcoxon

Examples

```
## Equal variance, small sample size
power.t.test(n = 5, power = 0.8)
sim.ssize.wilcox.test(rx = rnorm, ry = function(x) rnorm(x, mean = 2),
                      power = 0.8, n.min = 3, n.max = 10, step.size = 1)
sim.power.wilcox.test(nx = 6, rx = rnorm, rx.H0 = rnorm,
                      ny = 6, ry = function(x) rnorm(x, mean = 2),
                      ry.H0 = rnorm)
```

---

sim.ssize.wilcox.test *Sample Size for Wilcoxon Rank Sum and Signed Rank Tests*

Description

Simulate the empirical power of Wilcoxon rank sum and signed rank tests for computing the required sample size.

Usage

```
sim.ssize.wilcox.test(rx, ry = NULL, mu = 0, sig.level = 0.05, power = 0.8,
                      type = c("two.sample", "one.sample", "paired"),
                      alternative = c("two.sided", "less", "greater"),
                      n.min = 10, n.max = 200, step.size = 10,
                      iter = 10000, BREAK = TRUE)
```

Arguments

rx	function to simulate the values of x, respectively x-y in the paired case.
ry	function to simulate the values of y in the two-sample case
mu	true values of the location shift for the null hypothesis.
sig.level	significance level (Type I error probability)
power	two-sample, one-sample or paired test
type	one- or two-sided test. Can be abbreviated.
alternative	one- or two-sided test. Can be abbreviated.
n.min	integer, start value of grid search.
n.max	integer, stop value of grid search.
step.size	integer, step size used in the grid search.
iter	integer, number of interations of the simulations.
BREAK	logical, grid search stops when the emperical power is larger than the requested power.

Details

Functions `rx` and `ry` are used to simulate the data and functions `row_wilcoxon_twosample` and `row_wilcoxon_onesample` are used to efficiently compute the p values of the respective test.

We recommend a two steps procedure: In the first step, start with a wide grid and find out in which range of sample size values the intended power will be achieved. In the second step, the interval identified in the first step is used to find the sample size that leads to the required power setting `step.size = 1` and `BREAK = FALSE`. This approach is applied in the examples below.

Value

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

Author(s)

Matthias Kohl <Matthias.Kohl@stamats.de>

References

Wilcoxon, F (1945). Individual Comparisons by Ranking Methods. *Biometrics Bulletin*, **1**, 80-83.

See Also

`wilcox.test`, `wilcoxon`

Examples

```
##############################################################################
## two-sample
## iter = 1000 to reduce check time
##############################################################################
rx <- function(n) rnorm(n, mean = 0, sd = 1)
ry <- function(n) rnorm(n, mean = 0.5, sd = 1)
sim.ssize.wilcox.test(rx = rx, ry = ry, n.max = 100, iter = 1000)
sim.ssize.wilcox.test(rx = rx, ry = ry, n.min = 65, n.max = 70, step.size = 1,
                      iter = 1000, BREAK = FALSE)
## compared to
power.t.test(delta = 0.5, power = 0.8)

rx <- function(n) rnorm(n, mean = 0, sd = 1)
ry <- function(n) rnorm(n, mean = 0.5, sd = 1.5)
sim.ssize.wilcox.test(rx = rx, ry = ry, n.max = 100, iter = 1000, alternative = "less")
sim.ssize.wilcox.test(rx = rx, ry = ry, n.min = 85, n.max = 90, step.size = 1,
                      iter = 1000, BREAK = FALSE, alternative = "less")
## compared to
power.welch.t.test(delta = 0.5, sd = 1, sd2 = 1.5, power = 0.8, alternative = "one.sided")

rx <- function(n) rnorm(n, mean = 0.5, sd = 1)
ry <- function(n) rnorm(n, mean = 0, sd = 1)
sim.ssize.wilcox.test(rx = rx, ry = ry, n.max = 100, iter = 1000, alternative = "greater")
```

```
sim.ssize.wilcox.test(rx = rx, ry = ry, n.min = 50, n.max = 55, step.size = 1,
                      iter = 1000, BREAK = FALSE, alternative = "greater")
## compared to
power.t.test(delta = 0.5, power = 0.8, alternative = "one.sided")

rx <- function(n) rgamma(n, scale = 10, shape = 1)
ry <- function(n) rgamma(n, scale = 15, shape = 1)
sim.ssize.wilcox.test(rx = rx, ry = ry, n.max = 200, iter = 1000)
sim.ssize.wilcox.test(rx = rx, ry = ry, n.min = 125, n.max = 135, step.size = 1,
                      iter = 1000, BREAK = FALSE)

################################################################################
## one-sample
## iter = 1000 to reduce check time
################################################################################
rx <- function(n) rnorm(n, mean = 0.5, sd = 1)
sim.ssize.wilcox.test(rx = rx, mu = 0, type = "one.sample", n.max = 100, iter = 1000)
sim.ssize.wilcox.test(rx = rx, mu = 0, type = "one.sample", n.min = 33, n.max = 38,
                      step.size = 1, iter = 1000, BREAK = FALSE)
## compared to
power.t.test(delta = 0.5, power = 0.8, type = "one.sample")

sim.ssize.wilcox.test(rx = rx, mu = 0, type = "one.sample", n.max = 100, iter = 1000,
                      alternative = "greater")
sim.ssize.wilcox.test(rx = rx, mu = 0, type = "one.sample", n.min = 25, n.max = 30,
                     step.size = 1, iter = 1000, BREAK = FALSE, alternative = "greater")
## compared to
power.t.test(delta = 0.5, power = 0.8, type = "one.sample", alternative = "one.sided")

sim.ssize.wilcox.test(rx = rx, mu = 1, type = "one.sample", n.max = 100, iter = 1000,
                      alternative = "less")
sim.ssize.wilcox.test(rx = rx, mu = 1, type = "one.sample", n.min = 20, n.max = 30,
                      step.size = 1, iter = 1000, BREAK = FALSE, alternative = "less")
## compared to
power.t.test(delta = 0.5, power = 0.8, type = "one.sample", alternative = "one.sided")

rx <- function(n) rgamma(n, scale = 10, shape = 1)
sim.ssize.wilcox.test(rx = rx, mu = 5, type = "one.sample", n.max = 200, iter = 1000)
sim.ssize.wilcox.test(rx = rx, mu = 5, type = "one.sample", n.min = 40, n.max = 50,
                      step.size = 1, iter = 1000, BREAK = FALSE)

################################################################################
## paired
## identical to one-sample, requires random number generating function
## that simulates the difference x-y
## iter = 1000 to reduce check time
################################################################################
rxy <- function(n) rnorm(n, mean = 0.5, sd = 1)
sim.ssize.wilcox.test(rx = rxy, mu = 0, type = "paired", n.max = 100,
                      iter = 1000)
sim.ssize.wilcox.test(rx = rxy, mu = 0, type = "paired", n.min = 33,
                      n.max = 38, step.size = 1, iter = 1000, BREAK = FALSE)
## compared to
```

```
power.t.test(delta = 0.5, power = 0.8, type = "paired")
```

`ssize.pcc` *Sample Size Planning for Developing Classifiers Using High Dimensional Data*

Description

Calculate sample size for training set in developing classifiers using high dimensional data. The calculation is based on the probability of correct classification (PCC).

Usage

```
ssize.pcc(gamma, stdFC, prev = 0.5, nrFeatures, sigFeatures = 20, verbose = FALSE)
```

Arguments

`gamma`	tolerance between PCC(infty) and PCC(n).
`stdFC`	expected standardized fold-change; that is, expected fold-change devided by within class standard deviation.
`prev`	expected prevalence.
`nrFeatures`	number of features (variables) considered.
`sigFeatures`	number of significatn features; default (20) should be sufficient for most if not all cases.
`verbose`	print intermediate results.

Details

The computations are based the algorithm provided in Section~4.2 of Dobbin and Simon (2007). Prevalence is incorporated by the simple rough approach given in Section~4.4 (ibid.).

The results for prevalence equal to $50%$ are identical to the numbers computed by https://brb.nci.nih.gov/brb/samplesize/samplesize4GE.html. For other prevalences the numbers differ and are larger for our implementation.

Value

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

Note

`optimize` is used to solve equation (4.3) of Dobbin and Simon (2007), so you may see errors from it.

Author(s)

Matthias Kohl <Matthias.Kohl@stamats.de>

References

K. Dobbin and R. Simon (2007). Sample size planning for developing classifiers using high-dimensional DNA microarray data. *Biostatistics*, **8**(1):101-117.

K. Dobbin, Y. Zhao, R. Simon (2008). How Large a Training Set is Needed to Develop a Classifier for Microarray Data? *Clin Cancer Res.*, **14**(1):108-114.

See Also

optimize

Examples

```
## see Table 2 of Dobbin et al. (2008)
g <- 0.1
fc <- 1.6
ssize.pcc(gamma = g, stdFC = fc, nrFeatures = 22000)

## see Table 3 of Dobbin et al. (2008)
g <- 0.05
fc <- 1.1
ssize.pcc(gamma = g, stdFC = fc, nrFeatures = 22000)
```

---

ssize.propCI *Sample Size Calculation for Confidence Interval of a Proportion*

Description

Compute the sample size for the two-sided confidence interval of a single proportion.

Usage

```
ssize.propCI(prop, width, conf.level = 0.95,  method = "wald-cc")
```

Arguments

prop	expected proportion
width	width of the confidence interval
conf.level	confidence level
method	method used to compute the confidence interval; see Details.

Details

The computation is based on the asymptotic formulas provided in Section 2.5.2 of Fleiss et al. (2003). If `method = "wald-cc"` a continuity correction is applied.

There are also methods for Jeffreys, Clopper-Pearson, Wilson and the Agresti-Coull interval; see also `binomCI`.

Value

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

Author(s)

Matthias Kohl `<Matthias.Kohl@stamats.de>`

References

J.L. Fleiss, B. Levin and M.C. Paik (2003). *Statistical Methods for Rates and Proportions*. Wiley Series in Probability and Statistics.

W.W. Piegorsch (2004). Sample sizes for improved binomial confidence intervals. *Computational Statistics \& Data Analysis*, **46**, 309-316.

M. Thulin (2014). The cost of using exact confidence intervals for a binomial proportion. *Electronic Journal of Statistics*, **8**(1), 817-840.

See Also

`power.prop1.test`, `binomCI`

Examples

```
ssize.propCI(prop = 0.1, width = 0.1)
ssize.propCI(prop = 0.3, width = 0.1)
ssize.propCI(prop = 0.3, width = 0.1, method = "wald")
ssize.propCI(prop = 0.3, width = 0.1, method = "jeffreys")
ssize.propCI(prop = 0.3, width = 0.1, method = "clopper-pearson")
ssize.propCI(prop = 0.3, width = 0.1, method = "wilson")
ssize.propCI(prop = 0.3, width = 0.1, method = "agresti-coull")
```

---

volcano *Volcano Plots*

Description

Produce volcano plot(s) for simulations of power and type-I-error of tests.

Usage

```
## S3 method for class 'sim.power.ttest'
volcano(x, alpha = 1, shape = 19,
                                    hex = FALSE, bins = 50, ...)

## S3 method for class 'sim.power.wtest'
volcano(x, alpha = 1, shape = 19,
                                    hex = FALSE, bins = 50, ...)
```

Arguments

x	object of class sim.power.ttest.
alpha	bleding factor (default: no blending.
shape	point shape used.
hex	logical, should hexagonal binning be used.
bins	number of bins used for hexagonal binning.
...	further arguments that may be passed through).

Details

The plot generates a ggplot2 object that is shown.

Missing values are handled by the ggplot2 functions.

Value

Object of class gg and ggplot.

Author(s)

Matthias Kohl <Matthias.Kohl@stamats.de>

References

Wikipedia contributors, *Volcano plot (statistics)*, Wikipedia, The Free Encyclopedia, https://en.wikipedia.org/w/index.php?title=Volcano_plot_(statistics)&oldid=900217316 (accessed December 25, 2019).

For more sophisticated and flexible volcano plots see for instance: Blighe K, Rana S, Lewis M (2019). EnhancedVolcano: Publication-ready volcano plots with enhanced colouring and labeling. R/Bioconductor package. https://github.com/kevinblighe/EnhancedVolcano.

See Also

volcano

Examples

```
res1 <- sim.power.t.test(nx = 5, rx = rnorm, rx.H0 = rnorm,
                         ny = 10, ry = function(x) rnorm(x, mean = 3, sd = 3),
                         ry.H0 = function(x) rnorm(x, sd = 3))
volcano(res1)

## low number of iterations to reduce computation time
res2 <- sim.power.wilcox.test(nx = 6, rx = rnorm, rx.H0 = rnorm,
                       ny = 6, ry = function(x) rnorm(x, mean = 2),
                       ry.H0 = rnorm, iter = 100, conf.int = TRUE)
volcano(res2)
```

Index