Package 'pwrss'

December 14, 2022

Version 0.2.0

Type Package

Title Power and Sample Size Calculation Tools

Date 2022-12-13

Description Statistical power and minimum required sample size calculations for (1) testing a proportion (one-sample) against a constant, (2) testing a mean (one-sample) against a constant, (3) testing difference between two proportions (independent samples), (4) testing difference between two means (independent and paired samples), (5) testing a correlation (one-sample) against a constant, (6) testing difference between two correlations (independent samples), (7) testing a coefficient against a constant in multiple linear regression, (8) testing an indirect effect in the mediation analysis (Sobel, Joint, and Monte Carlo), (9) testing an R-squared against zero in linear regression, (10) testing an R-squared difference against zero in hierarchical regression, (11) testing an eta-squared or f-squared (for main and interaction effects) against zero in analysis of variance (could be one-way, two-way, and three-way), (12) testing an eta-squared or f-squared (for main and interaction effects) against zero in analysis of covariance (could be one-way, two-way, and three-way), (13) testing an eta-squared or f-squared (for between, within, and interaction effects) against zero in one-way repeated measures analysis of variance (with non-sphericity correction and repeated measures correlation). Alternative hypothesis can be formulated as ``not equal'', ``less'', ``greater'', ``non-inferior'', ``superior'', or ``equivalent'' in (1), (2), (3), and (4); as ``not equal'', ``less'', or ``greater'' in (5) and (6); but always as ``greater'' in (7), (8), (9), (10), and (11). Reference: Bulus & Polat (2022) <https://edarxiv.org/tfyxq/>.

Suggests knitr, rmarkdown

VignetteBuilder knitr

ByteCompile yes

LazyLoad yes

License GPL (>= 3)

Maintainer Metin Bulus <bulusmetin@gmail.com>

NeedsCompilation no

Author Metin Bulus [aut, cre]

Repository CRAN

Date/Publication 2022-12-14 08:10:11 UTC

R topics documented:

`plot` *Type I and Type II Error Plots for t, z, and F Tests*

Description

Plots Type I (alpha) and Type II (beta) errors for t, z, and F tests.

Usage

```
## S3 method for class 'pwrss'
plot(x, ...)
```

Arguments

`x`	an object of the type "pwrss" returned from one of the `pwrss` functions
`...`	for S3 generic/method consistency

Value

no return value at the moment

Examples

```
design <- pwrss.f.ancova(n.levels = c(3,3),
                n = 50, eta2 = 0.10)
plot(design)
```

power.f.test — *Statistical Power for the Generic F Test with Type I and Type II Error Plots*

Description

Calculates statistical power for the generic F test with Type I and Type II error plots.

Usage

```
power.f.test(ncp, df1, df2, alpha,
             plot = TRUE, plot.main = NULL, plot.sub = NULL)
```

Arguments

ncp	non-centrality parameter (lambda)
alpha	probability of type I error
df1	numerator degrees of freedom for the F test
df2	denominator degrees of freedom for the F test
plot	if TRUE plots Type I and Type II error plots
plot.main	title
plot.sub	subtitle

Value

power	statistical power $(1 - \beta)$

Examples

```
power.f.test(ncp = 1, df1 = 4, df2 = 100, alpha = 0.05)
```

power.t.test — *Statistical Power for the Generic t Test with Type I and Type II Error Plots*

Description

Calculates statistical power for the generic t test with Type I and Type II error plots.

Usage

```
power.t.test(ncp, df, alpha, alternative,
             plot = TRUE, plot.main = NULL, plot.sub = NULL)
```

Arguments

ncp	non-centrality parameter (lambda)
df	degrees of freedom for the t test
alpha	probability of type I error
alternative	direction or type of the hypothesis test: "not equal", "greater", "less", "equivalent", "non-inferior", or "superior". This information is needed only for power calculation
plot	if TRUE plots Type I and Type II error plots
plot.main	title
plot.sub	subtitle

Value

power	statistical power $(1 - \beta)$

Examples

```
power.t.test(ncp = 2.2, df = 100, alpha = 0.05, alternative = "not equal")
```

---

power.z.test *Statistical Power for the Generic z Test with Type I and Type II Error Plots*

Description

Calculates statistical power for the generic z test with Type I and Type II error plots.

Usage

```
power.z.test(ncp, alpha, alternative,
             plot = TRUE, plot.main = NULL, plot.sub = NULL)
```

Arguments

ncp	non-centrality parameter (lambda)
alpha	probability of type I error
alternative	direction or type of the hypothesis test: "not equal", "greater", "less", "equivalent", "non-inferior", or "superior". This information is needed only for power calculation
plot	if TRUE plots Type I and Type II error plots
plot.main	title
plot.sub	subtitle

Value

power	statistical power $(1 - \beta)$

Examples

```
power.z.test(ncp = 2.2, alpha = 0.05, alternative = "not equal")
```

---

pwrss.f.ancova — *Analysis of Variance (ANOVA) or Analysis of Covariance (ANCOVA) (F test)*

Description

Calculates statistical power or minimum required sample size for One-way/Two-way/Three-way Analysis of Variance (ANOVA) and Analysis of Covariance (ANCOVA). Set `n.covariates = 0` for ANOVA, and `n.covariates > 0` for ANCOVA. Note that in each case, the effect size (`eta2` or `f2`) should be obtained from the relevant model.

Usage

```
pwrss.f.ancova(eta2 = 0.01, f2 = eta2 / (1 - eta2),
               n.way = length(n.levels),
               n.levels = 2, n.covariates = 0, alpha = 0.05,
               n = NULL, power = NULL, verbose = TRUE)
```

Arguments

eta2	Expected Eta-squared
f2	Cohen's f2 (an alternative to `eta2` specification). f2 = eta2 / (1 - eta2)
n.way	1 for one-way, 2 for two-way, 3 for three-way ANOVA or ANCOVA. The default takes its value from the length of `n.levels`
n.levels	number of levels (groups) in each factor. For two factors each having two levels (groups) use e.g. c(2,2), for three factors each having two levels (groups) use e.g. c(2,2,2)
n.covariates	number of covariates in the ANCOVA model
n	total sample size
power	statistical power $(1 - \beta)$
alpha	probability of type I error
verbose	if `FALSE` no output is printed on the console. Useful for simulation, plotting, and whatnot

Value

parms	list of parameters used in calculation
test	type of the statistical test (z, t or F?)
df1	numerator degrees of freedom
df2	denominator degrees of freedom
ncp	non-centrality parameter
power	statistical power $(1 - \beta)$
n	sample size

References

Add references.

Examples

```
# one-way ANOVA
pwrss.f.ancova(n.levels = 2,
               n = 26, eta2 = 0.10)
pwrss.f.ancova(n.levels = 2,
               power = .80, eta2 = 0.10)

# two-way ANOVA
pwrss.f.ancova(n.levels = c(2,3),
               n = 26, eta2 = 0.10)
pwrss.f.ancova(n.levels = c(2,3),
               power = .80, eta2 = 0.10)

# three-way ANOVA
pwrss.f.ancova(n.levels = c(2,3,2),
               n = 26, eta2 = 0.10)
pwrss.f.ancova(n.levels = c(2,3,2),
               power = .80, eta2 = 0.10)

# one-way ANCOVA
pwrss.f.ancova(n.levels = 2, n.covariates = 1,
               n = 26, eta2 = 0.08)
pwrss.f.ancova(n.levels = 2, n.covariates = 1,
               power = .80, eta2 = 0.08)

# two-way ANCOVA
pwrss.f.ancova(n.levels = c(2,3), n.covariates = 1,
               n = 26, eta2 = 0.10)
pwrss.f.ancova(n.levels = c(2,3), n.covariates = 1,
               power = .80, eta2 = 0.08)

# three-way ANCOVA
pwrss.f.ancova(n.levels = c(2,3,2), n.covariates = 1,
               n = 26, eta2 = 0.10)
pwrss.f.ancova(n.levels = c(2,3,2), n.covariates = 1,
               power = .80, eta2 = 0.08)
```

`pwrss.f.reg` *R-squared Deviation from 0 (zero) or R-squared Difference in Hierarchical Multiple Linear Regression (F Test)*

Description

Calculates statistical power or minimum required sample size (only one can be NULL at a time) to test R-squared Deviation from 0 (zero) or to test R-squared difference from 0 (zero) in hierarchical regression.

Usage

```
pwrss.f.reg(r2 = 0.10, f2 = r2 /(1 - r2),
            k = 4, m = k, alpha = 0.05,
            n = NULL, power = NULL, verbose = TRUE)
```

Arguments

`r2`	Expected R-squared
`f2`	Cohen's f-squared (an alternative to `r2` specification). f2 = r2 / (1 - r2)
`k`	(total) number of predictors
`m`	(number of predictors in the subset of interest
`n`	sample size
`power`	statistical power $(1 - \beta)$
`alpha`	probability of type I error
`verbose`	if `FALSE` no output is printed on the console. Useful for simulation, plotting, and whatnot

Value

`parms`	list of parameters used in calculation
`test`	type of the statistical test (z, t or F?)
`df1`	numerator degrees of freedom
`df2`	denominator degrees of freedom
`ncp`	non-centrality parameter
`power`	statistical power $(1 - \beta)$
`n`	sample size

References

Add references.

Examples

```
# R-squared deviation from 0 (zero)
pwrss.f.reg(r2 = 0.20, k = 5, n = 26)
pwrss.f.reg(r2 = 0.20, k = 5, power = 0.80)

# Model 1 = 3 predictors
# Model 2 = 2 predictors
# k = 5 (total number of predictors)
# m = 2 (predictors whose incrimental R-squared change to be inspected)
pwrss.f.reg(r2 = 0.15, k = 5, m = 2, n = 26)
pwrss.f.reg(r2 = 0.15, k = 5, m = 2, power = 0.80)
```

---

pwrss.f.rmanova *Repeated Measures Analysis of Variance (RM-ANOVA) (F test)*

Description

Calculates statistical power or minimum required sample size for one-way Repeated Measures Analysis of Variance (RM-ANOVA).

Usage

```
pwrss.f.rmanova(eta2 = 0.10, f2 = eta2/(1 - eta2),
                repmeasures.r = 0.50, n.levels = 2, n.measurements = 2,
                epsilon = 1, alpha = 0.05,
                type = c("between","within","interaction"),
                n = NULL, power = NULL, verbose = TRUE)
```

Arguments

eta2	Expected Eta-squared
f2	Cohen's f2 (an alternative to eta2 specification). f2 = eta2 / (1 - eta2)
repmeasures.r	correlation between repeated measures
n.levels	number of levels (groups)
n.measurements	number of measurements
epsilon	non-sperhicity correction f, default 1 (means no violation of sphericity). Lower bound for epsilon = 1 / (n.measurements - 1)
n	total sample size
power	statistical power $(1 - \beta)$
alpha	probability of type I error
type	the effect to be tested: "between", "within", or "interaction"
verbose	if FALSE no output is printed on the console. Useful for simulation, plotting, and whatnot

Value

parms	list of parameters used in calculation
test	type of the statistical test (z, t or F?)
df1	numerator degrees of freedom
df2	denominator degrees of freedom
ncp	non-centrality parameter
power	statistical power $(1 - \beta)$
n	sample size

References

Add references.

Examples

```
pwrss.f.rmanova(eta2 = 0.10,  n.levels = 2, n.measurements = 3,
                type = "between", n = 10)
pwrss.f.rmanova(eta2 = 0.10,  n.levels = 2, n.measurements = 3,
                type = "between", power = 0.80)

pwrss.f.rmanova(eta2 = 0.10,  n.levels = 2, n.measurements = 3,
                type = "within", n = 10)
pwrss.f.rmanova(eta2 = 0.10,  n.levels = 2, n.measurements = 3,
                type = "within", power = 0.80)

pwrss.f.rmanova(eta2 = 0.10,  n.levels = 2, n.measurements = 3,
                type = "interaction", n = 10)
pwrss.f.rmanova(eta2 = 0.10,  n.levels = 2, n.measurements = 3,
                type = "interaction", power = 0.80)
```

---

pwrss.t.2means *Difference between Two Means (Independent Samples t Test)*

Description

Calculates statistical power or minimum required sample size (only one can be NULL at a time) to test difference between two means. For standardized mean difference (Cohen's d) set mu1 = d and use defaults for mu2, sd1, and sd2. If pooled standard deviation (psd) is available set sd1 = psd.

Usage

```
pwrss.t.2means(mu1, mu2 = 0, margin = 0,
               sd1 = ifelse(paired, sqrt(1/(2*(1-paired.r))), 1), sd2 = sd1,
               kappa = 1, paired = FALSE, paired.r = 0.50,
               alpha = 0.05, welch.df = FALSE,
               alternative = c("not equal","greater","less",
```

```
                          "equivalent","non-inferior","superior"),
       n2 = NULL, power = NULL, verbose = TRUE)
```

Arguments

mu1	expected mean in the first group
mu2	expected mean in the second group
sd1	standard devation in the first group
sd2	standard devation in the second group
kappa	n1/n2
paired	if TRUE paired samples t test
paired.r	correlation between the two repeated measures (e.g., pretest and posttest)
n2	sample size in the second group (or for the single group in paired samples)
power	statistical power $(1 - \beta)$
alpha	probability of type I error
welch.df	if TRUE uses Welch degrees of freedom adjustment when groups sizes or variances are not equal (applies to independent samples t test)
margin	non-inferority, superiority, or equivalence margin (margin = mu1 - mu2)
alternative	direction or type of the hypothesis test: "not equal", "greater", "less", "equivalent", "non-inferior", or "superior"
verbose	if FALSE no output is printed on the console. Useful for simulation, plotting, and whatnot

Value

parms	list of parameters used in calculation
test	type of the statistical test (z, t or F?)
df	degrees of freedom
ncp	non-centrality parameter
power	statistical power $(1 - \beta)$
n	sample size

References

Add references.

Examples

```
pwrss.t.2means(mu1 = 0.40, mu2 = 0.20,
               n2 = 474,  alternative = "greater")
pwrss.t.2means(mu1 = 0.40, mu2 = 0.20,
               power = 0.80,  alternative = "greater")
pwrss.t.2means(mu1 = 0.40, mu2 = 0.20, paired = TRUE,
```

```
               n2 = 474,  alternative = "greater")
pwrss.t.2means(mu1 = 0.40, mu2 = 0.20, paired = TRUE,
               power = 0.80,  alternative = "greater")

pwrss.t.2means(mu1 = 0.40, mu2 = 0.20,
               n2 = 474,  alternative = "not equal")
pwrss.t.2means(mu1 = 0.40, mu2 = 0.20,
               power = 0.80,  alternative = "not equal")
pwrss.t.2means(mu1 = 0.40, mu2 = 0.20, paired = TRUE,
               n2 = 474,  alternative = "not equal")
pwrss.t.2means(mu1 = 0.40, mu2 = 0.20, paired = TRUE,
               power = 0.80,  alternative = "not equal")


pwrss.t.2means(mu1 = 0.40, mu2 = 0.20,
               n2 = 474,  margin = -0.10, alternative = "non-inferior")
pwrss.t.2means(mu1 = 0.40, mu2 = 0.20,
               power = 0.80,  margin = -0.10, alternative = "non-inferior")
pwrss.t.2means(mu1 = 0.40, mu2 = 0.20, paired = TRUE,
               n2 = 474,  margin = -0.10, alternative = "non-inferior")
pwrss.t.2means(mu1 = 0.40, mu2 = 0.20, paired = TRUE,
               power = 0.80,  margin = -0.10, alternative = "non-inferior")


pwrss.t.2means(mu1 = 0.40, mu2 = 0.20,
               n2 = 474,  margin = 0.10, alternative = "superior")
pwrss.t.2means(mu1 = 0.40, mu2 = 0.20,
               power = 0.80,  margin = 0.10, alternative = "superior")
pwrss.t.2means(mu1 = 0.40, mu2 = 0.20, paired = TRUE,
               n2 = 474,  margin = 0.10, alternative = "superior")
pwrss.t.2means(mu1 = 0.40, mu2 = 0.20, paired = TRUE,
               power = 0.80,  margin = 0.10, alternative = "superior")


pwrss.t.2means(mu1 = 0.40, mu2 = 0.20,
               power = 0.80,  margin = 0.10, alternative = "equivalent")
pwrss.t.2means(mu1 = 0.40, mu2 = 0.20, paired = TRUE,
               power = 0.80,  margin = 0.10, alternative = "equivalent")
```

`pwrss.t.reg` *Test of a Single Coefficient in Multiple Linear Regression (t Test or z Test)*

Description

Calculates statistical power or minimum required sample size (only one can be NULL at a time) to test a single coefficient in multiple linear regression. The predictor is assumed to be continuous by default. However, one can find statistical power or minimum required sample size for a binary predictor (such as treatment and control groups in experimental designs) by specifying `sdx = sqrt(p*(1-p))` wherep is the proportion of subjects in one of the groups. The sample size in each group would be `n*p` and `n*(1-p)`.

Usage

```
pwrss.t.reg(beta = 0.25, beta0 = 0,
            sdx = 1, sdy = 1,
            k = 1, r2 = (beta * sdx / sdy)^2,
            alpha = 0.05, n = NULL, power = NULL,
            alternative = c("not equal", "less", "greater"),
            verbose = TRUE)

pwrss.z.reg(beta = 0.25, beta0 = 0,
            sdx = 1, sdy = 1,
            k = 1, r2 = (beta * sdx / sdy)^2,
            alpha = 0.05, n = NULL, power = NULL,
            alternative = c("not equal", "less", "greater"),
            verbose = TRUE)
```

Arguments

beta	expected regression coefficient. One can use standardized regression coefficient, but should use sdx = 1 and sdy = 1 or leave them out as they are default specifications
beta0	regression coefficient under null hypothesis
sdx	standard deviation of the predictor. For a binary predictor, sdx = sqrt(p*(1-p)) wherep is the proportion of subjects in one of the groups
sdy	standard deviation of the outcome
k	(total) number of predictors
r2	expected model R-squared. The default is r2 = (beta * sdx / sdy)^2 assuming a linear regression with one predictor. Thus, an r2 below this value will throw a warning. To consider other covariates in the model provide a value greater than the default along with argument k>1.
n	sample size
power	statistical power $(1 - \beta)$
alpha	probability of type I error
alternative	direction or type of the hypothesis test: "not equal", "greater", "less", "equivalent"
verbose	if FALSE no output is printed on the console. Useful for simulation, plotting, and whatnot

Value

parms	list of parameters used in calculation
test	type of the statistical test (z, t or F?)
df	numerator degrees of freedom
ncp	non-centrality parameter
power	statistical power $(1 - \beta)$
n	sample size

References

Add references.

Examples

```
# continuous predictor x
pwrss.t.reg(beta = 0.20,
            k = 5, r2 = 0.30,
            power = 0.80)
pwrss.t.reg(beta = 0.20,
            k = 5, r2 = 0.30,
            n = 140)

# binary predictor x
p <- 0.50 # proportion of subjects in one group
pwrss.t.reg(beta = 0.20, sdx = sqrt(p*(1-p)),
            k = 5, r2 = 0.30,
            power = 0.80)
pwrss.t.reg(beta = 0.20, sdx = sqrt(p*(1-p)) ,
            k = 5, r2 = 0.30,
            n = 552)
```

---

pwrss.z.2corrs *Difference between Two Correlations (Independent Samples z Test)*

Description

Calculates statistical power or minimum required sample size (only one can be NULL at a time) to test difference between two independent correlations.

Usage

```
pwrss.z.2corrs(r1 = 0.50, r2 = 0.30,
               alpha = 0.05, kappa = 1,
               alternative = c("not equal","greater","less"),
               n2 = NULL, power = NULL, verbose = TRUE)
```

Arguments

r1	expected correlation in the first group
r2	expected correlation in the second group
n2	sample size in the second group
kappa	n1/n2
power	statistical power $(1 - \beta)$
alpha	probability of type I error.
alternative	direction or type of the hypothesis test: "not equal", "greater", or "less"
verbose	if FALSE no output is printed on the console. Useful for simulation, plotting, and whatnot

Value

parms	list of parameters used in calculation
test	type of the statistical test (z, t or F?)
ncp	non-centrality parameter
power	statistical power $(1 - \beta)$
n	sample size

References

Add references.

Examples

```
pwrss.z.2corrs(r1 = .20, r2 = 0.30, n = 194)
pwrss.z.2corrs(r1 = .20, r2 = 0.30, power = .80)
```

---

pwrss.z.2means *Difference between Two Means (Independent Samples z Test)*

Description

Calculates statistical power or minimum required sample size (only one can be NULL at a time) to test difference between two means. For standardized mean difference (Cohen's d) set mu1 = d and use defaults for mu2, sd1, and sd2. If pooled standard deviation (psd) is available set sd1 = psd.

Usage

```
pwrss.z.2means(mu1, mu2 = 0, sd1 = 1, sd2 = sd1, margin = 0,
               kappa = 1, alpha = 0.05,
               alternative = c("not equal","greater","less",
                               "equivalent","non-inferior","superior"),
               n2 = NULL, power = NULL, verbose = TRUE)
```

Arguments

mu1	expected mean in the first group
mu2	expected mean in the second group
sd1	pooled standard devation in the first group
sd2	pooled standard devation in the second group
kappa	n1/n2
n2	sample size in the second group
power	statistical power $(1 - \beta)$
alpha	probability of type I error.

margin	non-inferority, superiority, or equivalence margin (margin = mu1 - mu2)
alternative	direction or type of the hypothesis test: "not equal", "greater", "less", "equivalent", "non-inferior", or "superior"
verbose	if FALSE no output is printed on the console. Useful for simulation, plotting, and whatnot

Value

parms	list of parameters used in calculation
test	type of the statistical test (z, t or F?)
ncp	non-centrality parameter
power	statistical power $(1 - \beta)$
n	sample size for the first and second group

References

Add references.

Examples

```
pwrss.z.2means(mu1 = 0.40, mu2 = 0.20,
               n2 = 474,  alternative = "greater")
pwrss.z.2means(mu1 = 0.40, mu2 = 0.20,
               power = 0.80,  alternative = "greater")

pwrss.z.2means(mu1 = 0.40, mu2 = 0.20,
               n2 = 474,  alternative = "not equal")
pwrss.z.2means(mu1 = 0.40, mu2 = 0.20,
               power = 0.80,  alternative = "not equal")


pwrss.z.2means(mu1 = 0.40, mu2 = 0.20,
               n2 = 474,  margin = -0.10, alternative = "non-inferior")
pwrss.z.2means(mu1 = 0.40, mu2 = 0.20,
               power = 0.80,  margin = -0.10, alternative = "non-inferior")

pwrss.z.2means(mu1 = 0.40, mu2 = 0.20,
               n2 = 474,  margin = 0.10, alternative = "superior")
pwrss.z.2means(mu1 = 0.40, mu2 = 0.20,
               power = 0.80,  margin = 0.10, alternative = "superior")


pwrss.z.2means(mu1 = 0.40, mu2 = 0.20,
               power = 0.80,  margin = 0.10, alternative = "equivalent")
```

pwrss.z.2props *Difference between Two Proportions (Independent Samples z Test)*

Description

Calculates statistical power or minimum required sample size (only one can be NULL at a time) to test difference between two proportions.

Usage

```
pwrss.z.2props(p1, p2, margin = 0, arcsin.trans = TRUE, kappa = 1, alpha = 0.05,
               alternative = c("not equal","greater","less",
                               "equivalent","non-inferior","superior"),
               n2 = NULL, power = NULL, verbose = TRUE)
```

Arguments

p1	expected proportion in the first group
p2	expected proportion in the second group
arcsin.trans	if TRUE uses arcsin transformation (default), if FALSE uses normal approximation
kappa	n1/n2
n2	sample size in the second group
power	statistical power $(1 - \beta)$
alpha	probability of type I error.
margin	non-inferority, superiority, or equivalence margin (margin = p1 - p2)
alternative	direction or type of the hypothesis test: "not equal", "greater", "less", "equivalent", "non-inferior", or "superior"
verbose	if FALSE no output is printed on the console. Useful for simulation, plotting, and whatnot

Value

parms	list of parameters used in calculation
test	type of the statistical test (z, t or F?)
ncp	non-centrality parameter
power	statistical power $(1 - \beta)$
n	sample size for the first and second group

References

Add references.

Examples

```
pwrss.z.2props(p1 = 0.65, p2 = 0.85,
               n2 = 55, alternative = "less")
pwrss.z.2props(p1 = 0.65, p2 = 0.85,
               power = 0.80, alternative = "less")

pwrss.z.2props(p1 = 0.65, p2 = 0.85,
               n2 = 55, alternative = "not equal")
pwrss.z.2props(p1 = 0.65, p2 = 0.85,
               power = 0.80, alternative = "not equal")

pwrss.z.2props(p1 = 0.65, p2 = 0.85, n2 = 55,
               margin = -.10, alternative = "non-inferior")
pwrss.z.2props(p1 = 0.65, p2 = 0.85, power = 0.80,
               margin = -.10, alternative = "non-inferior")

pwrss.z.2props(p1 = 0.65, p2 = 0.85, n2 = 55,
               margin = .10, alternative = "superior")
pwrss.z.2props(p1 = 0.65, p2 = 0.85, power = 0.80,
               margin = .10, alternative = "superior")

pwrss.z.2props(p1 = 0.65, p2 = 0.85, power = 0.80,
               margin = .10, alternative = "equivalent")
```

---

pwrss.z.corr *A Correlation against 0 (zero) (One Sample z Test)*

Description

Calculates statistical power or minimum required sample size (only one can be NULL at a time) to test a correlation against 0 (zero).

Usage

```
pwrss.z.corr(r = 0.50, r0 = 0, alpha = 0.05,
             alternative = c("not equal","greater","less"),
             n = NULL, power = NULL, verbose = TRUE)
```

Arguments

r	expected correlation
r0	constant proportion to be compared
n	sample size
power	statistical power $(1 - \beta)$
alpha	probability of type I error.
alternative	direction or type of the hypothesis test: "not equal", "greater", or "less"
verbose	if FALSE no output is printed on the console. Useful for simulation, plotting, and whatnot

Value

parms	list of parameters used in calculation
test	type of the statistical test (z, t or F?)
ncp	non-centrality parameter
power	statistical power $(1 - \beta)$
n	sample size

References

Add references.

Examples

```
pwrss.z.corr(r = .20, n = 194)
pwrss.z.corr(r = .20, power = .80, alternative = "not equal")
```

---

pwrss.z.mean *A Mean against a Constant (One Sample z Test)*

Description

Calculates statistical power or minimum required sample size (only one can be NULL at a time) to test a mean against a constant mean.

Usage

```
pwrss.z.mean(mu, sd = 1, mu0 = 0, margin = 0, alpha = 0.05,
             alternative = c("not equal","greater","less",
                             "equivalent","non-inferior","superior"),
             n = NULL, power = NULL, verbose = TRUE)
```

Arguments

mu	expected mean
sd	expected standard devation
mu0	constant to be tested against (a mean)
n	sample size
power	statistical power $(1 - \beta)$
alpha	probability of type I error.
margin	non-inferority, superiority, or equivalence margin (margin = p - p0)
alternative	direction or type of the hypothesis test: "not equal", "greater", "less", "equivalent", "non-inferior", or "superior"
verbose	if FALSE no output is printed on the console. Useful for simulation, plotting, and whatnot

Value

parms	list of parameters used in calculation
test	type of the statistical test (z, t or F?)
ncp	non-centrality parameter
power	statistical power $(1 - \beta)$
n	sample size

References

Add references.

Examples

```
pwrss.z.mean(mu = 0.40, mu0 = 0.20, sd = 1,
             n = 155, alternative = "greater")
pwrss.z.mean(mu = 0.40, mu0 = 0.20, sd = 1,
             power = 0.80, alternative = "greater")

pwrss.z.mean(mu = 0.40, mu0 = 0.20, sd = 1,
             n = 155, alternative = "not equal")
pwrss.z.mean(mu = 0.40, mu0 = 0.20, sd = 1,
             power = 0.80, alternative = "not equal")

pwrss.z.mean(mu = 0.40, mu0 = 0.20, sd = 1,
             n = 155, margin = -0.05, alternative = "non-inferior")
pwrss.z.mean(mu = 0.40, mu0 = 0.20, sd = 1,
             power = 0.80, margin = -0.05, alternative = "non-inferior")

pwrss.z.mean(mu = 0.40, mu0 = 0.20, sd = 1,
             n = 155, margin = 0.05, alternative = "superior")
pwrss.z.mean(mu = 0.40, mu0 = 0.20, sd = 1,
             power = 0.80, margin = 0.05, alternative = "superior")

pwrss.z.mean(mu = 0.40, mu0 = 0.20, sd = 1,
             power = 0.80, margin = 0.05, alternative = "equivalent")
```

---

pwrss.z.med — *Test of an Indirect Effect in the Mediation Analysis (z Test, Joint Test, and Monte Carlo)*

Description

Calculates statistical power or minimum required sample size (only one can be NULL at a time) to test indirect effects in mediation analysis. One can consider explanatory power of the covariates in the mediation and outcome model via specifying R-squared values accordingly.

Usage

```
pwrss.z.med(a, b, cp = 0,
            sdx = 1, sdm = 1, sdy = 1,
            r2m.x = a^2 * sdx^2 / sdm^2,
            r2y.mx = (b^2 * sdm^2 + cp^2 * sdx^2) / sdy^2,
            n = NULL, power = NULL, alpha = 0.05,
            alternative = c("not equal", "less", "greater"),
            mc = TRUE, nsims = 1000, ndraws = 1000,
            verbose = TRUE)
```

Arguments

a	expected regression coefficient for X -> M path. One can use standardized regression coefficient, but should use sdx = 1 and sdm = 1 or leave them out as they are default specifications
b	expected regression coefficient for M -> Y path. One can use standardized regression coefficient, but should use sdm = 1 and sdy = 1 or leave them out as they are default specifications
cp	expected regression coefficient for X -> Y path. One can use standardized regression coefficient, but should use sdx = 1 and sdy = 1 or leave them out as they are default specifications
sdx	standard deviation of the predictor (X). For a binary predictor, sdx = sqrt(p*(1-p)) wherep is the proportion of subjects in one of the groups
sdm	standard deviation of the mediator (M)
sdy	standard deviation of the outcome (Y)
r2m.x	expected R-squared value for the mediator model (M ~ X). The default is r2m.x = a^2 * sdx^2 / sdm^2 assuming that X is the only predictor. Thus, an r2m.x below this value will throw a warning. To consider other covariates in the model provide a value greater than the default.
r2y.mx	expected R-squared value for the outcome model (Y ~ M + X). The default is r2y.mx = (b^2 * sdm^2 + cp^2 * sdx^2) / sdy^2 assuming that M and X are the only predictors. Thus, an r2y.mx below this value will throw a warning. To consider other covariates in the model provide a value greater than the default.
n	sample size
power	statistical power $(1 - \beta)$
alpha	probability of type I error
alternative	direction or type of the hypothesis test: "not equal", "greater", "less", "equivalent"
mc	logical; TRUE for monte carlo simulation based power
nsims	number of replications, if mc = TRUE
ndraws	number of draws from the distribution of the path coefficients for each replication, if mc = TRUE
verbose	if FALSE no output is printed on the console. Useful for simulation, plotting, and whatnot

Value

parms	list of parameters used in calculation
test	type of the statistical test (z, t or F?)
ncp	non-centrality parameter
power	statistical power $(1 - \beta)$
n	sample size

References

Add references.

Examples

```
# with standardized coefficients
pwrss.z.med(a = 0.25, b = 0.25, cp = 0.10, n = 200)
pwrss.z.med(a = 0.25, b = 0.25, cp = 0.10, power = 0.80)

# with binary predictor X such as treatment/control variable
# in this case standardized coefficients for path a and cp would be Cohen's d values
p <- 0.50 # proportion of subjects in one group
pwrss.z.med(a = 0.25, b = 0.25, cp = 0.10,
            sdx = sqrt(p*(1-p)), n = 200)
pwrss.z.med(a = 0.25, b = 0.25, cp = 0.10,
            sdx = sqrt(p*(1-p)), power = 0.80)
```

---

pwrss.z.prop *A Proportion against a Constant (One Sample z Test)*

Description

Calculates statistical power or minimum required sample size (only one can be NULL at a time) to test a proportion against a constant proportion.

Usage

```
pwrss.z.prop(p, p0 = 0, margin = 0, arcsin.trans = TRUE, alpha = 0.05,
             alternative = c("not equal","greater","less",
                             "equivalent","non-inferior","superior"),
             n = NULL, power = NULL, verbose = TRUE)
```

Arguments

p	expected proportion
p0	constant to be tested against (a proportion)
arcsin.trans	if TRUE uses arcsin transformation (default), if FALSE uses normal approximation
n	sample size
power	statistical power $(1 - \beta)$
alpha	probability of type I error.
margin	non-inferority, superiority, or equivalence margin (margin = p - p0)
alternative	direction or type of the hypothesis test: "not equal", "greater", "less", "equivalent", "non-inferior", or "superior"
verbose	if FALSE no output is printed on the console. Useful for simulation, plotting, and whatnot

Value

parms	list of parameters used in calculation
test	type of the statistical test (z, t or F?)
ncp	non-centrality parameter
power	statistical power $(1 - \beta)$
n	sample size

References

Add references.

Examples

```
pwrss.z.prop(p = 0.05, p0 = 0.02, n = 190, power = NULL,
             alternative = "greater")
pwrss.z.prop(p = 0.05, p0 = 0.02, power = 0.80,
             alternative = "greater")

pwrss.z.prop(p = 0.05, p0 = 0.02, n = 190, power = NULL,
             alternative = "not equal")
pwrss.z.prop(p = 0.05, p0 = 0.02, power = 0.80,
             alternative = "not equal")

pwrss.z.prop(p = 0.05, p0 = 0.02, n = 190, power = NULL,
             margin = -0.005, alternative = "non-inferior")
pwrss.z.prop(p = 0.05, p0 = 0.02, power = 0.80,
             margin = -0.005, alternative = "non-inferior")

pwrss.z.prop(p = 0.05, p0 = 0.02, n = 190, power = NULL,
             margin = 0.005, alternative = "superior")
pwrss.z.prop(p = 0.05, p0 = 0.02, power = 0.80,
             margin = 0.005, alternative = "superior")
```

```
pwrss.z.prop(p = 0.05, p0 = 0.02, power = 0.80,
             margin = 0.005, alternative = "equivalent")
```

Index