Package ‘pwrss’

April 11, 2023

Version 0.3.1

Type Package

Title Statistical Power and Sample Size Calculation Tools

Date 2023-04-10

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 or groups (parametric and non-parametric tests for independent and paired samples), (5) testing a correlation (one-sample) against a constant, (6) testing difference between two correlations (independent samples), (7) testing a single coefficient in multiple linear regression, logistic regression, and Poisson regression (with standardized or unstandardized coefficients, with no covariates or covariate adjusted), (8) testing an indirect effect (with standardized or unstandardized coefficients, with no covariates or covariate adjusted) in the mediation analysis (Sobel, Joint, and Monte Carlo tests), (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), and (14) testing goodness-of-fit or independence for contingency tables. 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), (6), (7) and (8); but always as ``greater'' in (9), (10), (11), (12), (13), and (14). Reference: Bulus and Polat (2023) <https://osf.io/ua5fc>.

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 2023-04-11 21:00:02 UTC

R topics documented:

`plot` *Type I and Type II Error Plot*

Description

Plots Type I (alpha) and Type II (beta) errors for t, z, F, and Chi-square 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.chisq.test` *Statistical Power for the Generic Chi-square Test*

Description

Calculates statistical power for the generic chi-square test with (optional) Type I and Type II error plots. Unlike other more specific functions `power.chisq.test()` function allows multiple values for one parameter at a time (only when `plot = FALSE`).

Usage

```
power.chisq.test(ncp, df, alpha = 0.05, plot = TRUE,
                 plot.main = NULL, plot.sub = NULL,
                 verbose = TRUE)
```

Arguments

`ncp`	non-centrality parameter (lambda)
`df`	degrees of freedom. For example, for the test of homogeneity or independence df = (nrow - 1)*(ncol - 1)
`alpha`	probability of type I error
`plot`	if `TRUE` plots Type I and Type II error
`plot.main`	plot title
`plot.sub`	plot subtitle
`verbose`	if `FALSE` no output is printed on the console. Useful for simulation, plotting, and whatnot

Value

`power`	statistical power $(1 - \beta)$

Examples

```
# power is defined as the probability of observing Chi-square-statistics
# greater than the critical Chi-square value
power.chisq.test(ncp = 20, df = 100, alpha = 0.05)

# power of multiple Chi-square-statistics
power.chisq.test(ncp = c(5, 10, 15, 20), plot = FALSE,
                 df = 100, alpha = 0.05)
```

---

power.f.test *Statistical Power for the Generic F Test*

Description

Calculates statistical power for the generic F test with (optional) Type I and Type II error plots. Unlike other more specific functions `power.f.test()` function allows multiple values for one parameter at a time (only when `plot = FALSE`).

Usage

```
power.f.test(ncp, df1, df2, alpha = 0.05, plot = TRUE,
             plot.main = NULL, plot.sub = NULL,
             verbose = TRUE)
```

Arguments

`ncp`	non-centrality parameter (lambda)
`alpha`	probability of type I error
`df1`	numerator degrees of freedom
`df2`	denominator degrees of freedom
`plot`	if `TRUE` plots Type I and Type II error
`plot.main`	plot title
`plot.sub`	plot subtitle
`verbose`	if `FALSE` no output is printed on the console. Useful for simulation, plotting, and whatnot

Value

`power`	statistical power $(1 - \beta)$

Examples

```
# power is defined as the probability of observing F-statistics
# greater than the critical F value
power.f.test(ncp = 1, df1 = 4, df2 = 100, alpha = 0.05)

# power of multiple F-statistics
power.f.test(ncp = c(1.0, 1.5, 2.0, 2.5), plot = FALSE,
             df1 = 4, df2 = 100, alpha = 0.05)
```

`power.t.test` *Statistical Power for the Generic t Test*

Description

Calculates statistical power for the generic t test with (optional) Type I and Type II error plots. Unlike other more specific functions `power.t.test()` function allows multiple values for one parameter at a time (only when `plot = FALSE`).

Usage

```
power.t.test(ncp, df, alpha = 0.05,
             alternative = c("not equal", "greater", "less",
                             "non-inferior", "superior", "equivalent"),
             plot = TRUE, plot.main = NULL, plot.sub = NULL,
             verbose = TRUE)
```

Arguments

`ncp`	non-centrality parameter (lambda)
`df`	degrees of freedom
`alpha`	probability of type I error
`alternative`	direction or type of the hypothesis test: "not equal", "greater", "less", "equivalent", "non-inferior", or "superior". The same non-centrality parameters will produce the same power rates for "greater", "less", "non-inferior", and "superior" tests. Different labels have been used merely for consistency. However, it should be noted that the non-centrality parameter should conform to the specific test type
`plot`	if `TRUE` plots Type I and Type II error
`plot.main`	plot title
`plot.sub`	plot subtitle
`verbose`	if `FALSE` no output is printed on the console. Useful for simulation, plotting, and whatnot

Value

power statistical power $(1-\beta)$

Examples

```
# power is defined as the probability of observing t-statistics
# greater than the positive critical t value OR
# less than the negative critical t value
power.t.test(ncp = 1.96, df = 99, alpha = 0.05,
             alternative = "not equal")

# power is defined as the probability of observing t-statistics
# greater than the critical t value
power.t.test(ncp = 1.96, df = 99, alpha = 0.05,
             alternative = "greater")

# power is defined as the probability of observing t-statistics
# greater than the critical t value where the non-centrality parameter
# for the alternative distribution is adjusted for the non-inferiority margin
power.t.test(ncp = 1.98, df = 99, alpha = 0.05,
             alternative = "non-inferior")

# power is defined as the probability of observing t-statistics
# greater than the critical t value where the non-centrality parameter
# for the alternative distribution is adjusted for the superiority margin
power.t.test(ncp = 1.94, df = 99, alpha = 0.05,
             alternative = "superior")

# power is defined as the probability of observing t-statistics
# less than the positive critical t value AND
# greater than the negative critical t value
# the non-centrality parameter is for the null distribution
# and is derived from the equivalence margins (lower and upper)
power.t.test(ncp = 1.96, df = 999, alpha = 0.05,
             alternative = "equivalent")
# or, define lower and upper bound with rbind()
power.t.test(ncp = rbind(-1.96, 1.96),
             df = 999, alpha = 0.05,
             alternative = "equivalent")
```

---

power.z.test *Statistical Power for the Generic z Test*

Description

Calculates statistical power for the generic z test with (optional) Type I and Type II error plots. Unlike other more specific functions `power.z.test()` function allows multiple values for one parameter at a time (only when `plot = FALSE`).

Usage

```
power.z.test(ncp, alpha = 0.05,
             alternative = c("not equal", "greater", "less",
                             "non-inferior", "superior", "equivalent"),
             plot = TRUE, plot.main = NULL, plot.sub = NULL,
             verbose = TRUE)
```

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". The same non-centrality parameters will produce the same power rates for "greater", "less", "non-inferior", and "superior" tests. Different labels have been used for consistency. However, it should be noted that the non-centrality parameter should conform to the specific test type
plot	if TRUE plots Type I and Type II error
plot.main	plot title
plot.sub	plot subtitle
verbose	if FALSE no output is printed on the console. Useful for simulation, plotting, and whatnot

Value

power	statistical power $(1-\beta)$

Examples

```
# power defined as the probability of observing z-statistics
# greater than the positive critical t value OR
# less than the negative critical t value
power.z.test(ncp = 1.96, alpha = 0.05,
             alternative = "not equal")

# power is defined as the probability of observing z-statistics
# greater than the critical t value
power.z.test(ncp = 1.96, alpha = 0.05,
             alternative = "greater")

# power is defined as the probability of observing z-statistics
# greater than the critical t value where the non-centrality parameter
# for the alternative distribution is adjusted for the non-inferiority margin
power.z.test(ncp = 1.98, alpha = 0.05,
             alternative = "non-inferior")

# power is defined as the probability of observing z-statistics
# greater than the critical t value where the non-centrality parameter
```

```
# for the alternative distribution is adjusted for the superiority margin
power.z.test(ncp = 1.94, alpha = 0.05,
             alternative = "superior")

# power is defined as the probability of observing z-statistics
# less than the positive critical t value AND
# greater than the negative critical t value
# the non-centrality parameter is for the null distribution
# and is derived from the equivalence margins (lower and upper)
power.z.test(ncp = 1.96, alpha = 0.05,
             alternative = "equivalent")
# or, define lower and upper bound with rbind()
power.z.test(ncp = rbind(-1.96, 1.96), alpha = 0.05,
             alternative = "equivalent")
```

---

pwrss.chisq.gofit *Goodness-of-Fit or Independence (Chi-square Test)*

Description

Calculates statistical power or minimum required sample size (only one can be NULL at a time) for Chi-square goodness-of-fit or independence test.

Usage

```
pwrss.chisq.gofit(p1 = c(0.50, 0.50),
                  p0 = .chisq.fun(p1)$p0,
                  w = .chisq.fun(p1)$w,
                  df = .chisq.fun(p1)$df,
                  n = NULL, power = NULL,
                  alpha = 0.05, verbose = TRUE)
```

Arguments

p1	a vector or matrix of cell probabilities under alternative hypothesis
p0	a vector or matrix of cell probabilities under null hypothesis. Calculated automatically when p1 is specified. The default can be overwritten by the user via providing a vector of the same size or matrix of the same dimensions as p1
w	effect size. Computed from p1 and p0 automatically; however, it can be any of Cohen's W, Phi coefficient, Cramer's V, etc. when specified by the user. Phi coefficient is defined as sqrt(X2/n) and Cramer's V is defined as sqrt(X2/(n*v)) where v is min(nrow - 1, ncol - 1) and X2 is the chi-square statistic
df	degrees of freedom. Defined as (ncells - 1) if p1 is a vector, and as (nrows - 1) * (ncols - 1) if p1 is a matrix
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

Value

parms	list of parameters used in calculation
test	type of the statistical test (Chi-square test)
df	degrees of freedom
ncp	non-centrality parameter
power	statistical power $(1-\beta)$
n	total sample size

References

Cohen, J. (1988). Statistical power analysis for the behavioral sciences (2nd ed.). Lawrence Erlbaum Associates.

Examples

```
# ---------------------------------------------------------#
# Example 1: Cohen's W                                      #
# goodness-of-fit test for 1 x k or k x 1 table             #
# How many subjects are needed to claim that                #
# girls choose STEM related majors less than males?        #
# ---------------------------------------------------------#

## Option 1: Use cell probabilities
## from https://www.aauw.org/resources/research/the-stem-gap/
## 28 percent of the  workforce in STEM field is women
prob.mat <- c(0.28, 0.72) # null hypothesis states that c(0.50, 0.50)
pwrss.chisq.gofit(p1 = c(0.28, 0.72),
                  alpha = 0.05, power = 0.80)

## Option 2: Use Cohe's W = 0.44
## df is k - 1 for Cohen's W
pwrss.chisq.gofit(w = 0.44, df = 1,
                  alpha = 0.05, power = 0.80)


# ---------------------------------------------------------#
# Example 2: Phi Coefficient (or Cramer's V or Cohen's W)  #
# test of independence for 2 x 2 contingency tables        #
# How many subjects are needed to claim that                #
# girls are underdiagnosed with ADHD?                       #
# ---------------------------------------------------------#

## Option 1: Use cell probabilities
## from https://time.com/growing-up-with-adhd/
## 5.6 percent of girls and 13.2 percent of boys are diagnosed with ADHD
prob.mat <- rbind(c(0.056, 0.132),
                  c(0.944, 0.868))
colnames(prob.mat) <- c("Girl", "Boy")
rownames(prob.mat) <- c("ADHD", "No ADHD")
prob.mat
```

```
pwrss.chisq.gofit(p1 = prob.mat,
                  alpha = 0.05, power = 0.80)

## Option 2: Use Phi coefficient = 0.1302134
## df is 1 for Phi coefficient
pwrss.chisq.gofit(w = 0.1302134, df = 1,
                  alpha = 0.05, power = 0.80)


# ------------------------------------------------------#
# Example 3: Cramer's V (or Cohen's W)                  #
# test of independence for j x k contingency tables     #
# How many subjects are needed to detect the relationship #
# between depression severity and gender?               #
# ------------------------------------------------------#

## Option 1: Use cell probabilities
## from https://doi.org/10.1016/j.jad.2019.11.121
prob.mat <- cbind(c(0.6759, 0.1559, 0.1281, 0.0323, 0.0078),
                  c(0.6771, 0.1519, 0.1368, 0.0241, 0.0101))
rownames(prob.mat) <- c("Normal", "Mild", "Moderate", "Severe", "Extremely Severe")
colnames(prob.mat) <- c("Female", "Male")
prob.mat
pwrss.chisq.gofit(p1 = prob.mat,
                  alpha = 0.05, power = 0.80)

# Option 2: Use Cramer's V = 0.03022008 based on 5 x 2 contingency table
# df is (nrow - 1) * (ncol - 1) for Cramer's V
pwrss.chisq.gofit(w = 0.03022008, df = 4,
                  alpha = 0.05, power = 0.80)
```

`pwrss.f.ancova` *Analysis of (Co)Variance (F test)*

Description

Calculates statistical power or minimum required sample size for one-way, two-way, or three-way ANOVA/ANCOVA. Set `n.covariates = 0` for ANOVA, and `n.covariates > 0` for ANCOVA. Note that in each case, the effect size (partial) (`eta2` or `f2`) should be obtained from the relevant model.

Formulas are validated using Monte Carlo simulation, G*Power, and tables in PASS documentation.

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	expected 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 example, 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
alpha	probability of type I error
power	statistical power $(1-\beta)$
verbose	if FALSE no output is printed on the console

Value

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

References

Bulus, M., & Polat, C. (in press). pwrss R paketi ile istatistiksel guc analizi [Statistical power analysis with pwrss R package]. Ahi Evran Universitesi Kirsehir Egitim Fakultesi Dergisi. https://osf.io/ua5fc/download/

Cohen, J. (1988). Statistical power analysis for the behavioral sciences (2nd ed.). Lawrence Erlbaum Associates.

Examples

```
##############################################
#              one-way ANOVA                 #
##############################################

# a researcher is expecting a difference of
# Cohen's d = 0.50 between treatment and control
# translating into Eta-squared = 0.059

# estimate sample size using ANOVA approach
pwrss.f.ancova(eta2 = 0.059, n.levels = 2,
               alpha = 0.05, power = .80)
```

```
# estimate sample size using regression approach(F test)
pwrss.f.reg(r2 = 0.059, k = 1,
            alpha = 0.05, power = 0.80)

# estimate sample size using regression approach (t test)
p <- 0.50 # proportion of sample in treatment
pwrss.t.reg(beta1 = 0.50, r2 = 0,
            k = 1, sdx = sqrt(p*(1-p)),
            alpha = 0.05, power = 0.80)

# estimate sample size using t test approach
pwrss.t.2means(mu1 = 0.50,
               alpha = 0.05, power = 0.80)

#############################################
#              two-way ANOVA                #
#############################################

# a researcher is expecting a partial Eta-squared = 0.03
# for interaction of treatment (Factor A) with
# gender consisting of two levels (Factor B)

pwrss.f.ancova(eta2 = 0.03, n.levels = c(2,2),
               alpha = 0.05, power = 0.80)

# estimate sample size using regression approach (F test)
# one dummy for treatment, one dummy for gender, and their interaction (k = 3)
# partial Eta-squared is equivalent to the increase in R-squared by adding
# only the interaction term (m = 1)
pwrss.f.reg(r2 = 0.03, k = 3, m = 1,
            alpha = 0.05, power = 0.80)

#############################################
#              one-way ANCOVA               #
#############################################

# a researcher is expecting an adjusted difference of
# Cohen's d = 0.45 between treatment and control after
# controllling for the pretest (n.cov = 1)
# translating into Eta-squared = 0.048

pwrss.f.ancova(eta2 = 0.048, n.levels = 2, n.cov = 1,
               alpha = 0.05, power = .80)

#############################################
#              two-way ANCOVA               #
#############################################

# a researcher is expecting an adjusted partial Eta-squared = 0.02
# for interaction of treatment (Factor A) with
# gender consisting of two levels (Factor B)
```

```
pwrss.f.ancova(eta2 = 0.02, n.levels = c(2,2), n.cov = 1,
               alpha = 0.05, power = .80)
```

---

pwrss.f.reg *Linear Regression: R-squared or R-squared Difference (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) in linear regression or to test R-squared difference between two linear regression models. The test of R-squared difference is often used to evaluate incremental contribution of a set of predictors in hierarchical linear regression.

Formulas are validated using Monte Carlo simulation, G*Power, and tables in PASS documentation.

Usage

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

Arguments

`r2`	expected R-squared (or R-squared change)
`f2`	expected 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. By default `m = k`, which implies that one is interested in the contribution of all predictors, and tests whether R-squared value is different from 0 (zero)
`n`	sample size
`power`	statistical power $(1 - \beta)$
`alpha`	probability of type I error
`verbose`	if `FALSE` no output is printed on the console

Value

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

References

Bulus, M., & Polat, C. (in press). pwrss R paketi ile istatistiksel guc analizi [Statistical power analysis with pwrss R package]. Ahi Evran Universitesi Kirsehir Egitim Fakultesi Dergisi. https://osf.io/ua5fc/download/

Cohen, J. (1988). Statistical power analysis for the behavioral sciences (2nd ed.). Lawrence Erlbaum Associates.

Examples

```
# EXample 1: A researcher is expecting that
# three variables together explain 15 percent of the variance
# in the outcome (R-squared = 0.15).
pwrss.f.reg(r2 = 0.15, k = 3,
            alpha = 0.05, power = 0.80)

# Example 2: A researcher is expecting that
# adding two more variables will increase R-squared
# from 0.15 (with 3 predictors) to 0.20 (with 3 + 2 predictors)
# k = 5 (total number of predictors)
# m = 2 (predictors whose incremental contribution to R-squared change is of interest)
pwrss.f.reg(r2 = 0.05, k = 5, m = 2,
            alpha = 0.05, power = 0.80)
```

---

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

Description

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

Formulas are validated using Monte Carlo simulation, G*Power, and tables in PASS documentation.

Usage

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

Arguments

`eta2`	expected (partial) Eta-squared
`f2`	expected Cohen's f-squared (an alternative to `eta2` specification). `f2 = eta2 / (1 - eta2)`

`corr.rm`	expected correlation between repeated measures. For example, for pretest/posttest designs, this is the correlation between pretest and posttest scores regardless of group membership. The default is 0.50
`n.levels`	number of levels (groups). For example, for randomized controlled trials with two arms (treatment/control) it takes a value of 2
`n.rm`	number of measurements. For example, for pretest/posttest designs it takes a value of 2. When there is a follow-up test it takes a value of 3
`epsilon`	non-sperhicity correction factor, default is 1 (means no violation of sphericity). Lower bound for this argument is `epsilon` = 1 / (`n.rm` - 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". The type of the effect depends on the hypothesis test. If the interest is in the group effect after controlling for the time effect use "between"; if the interest is the time effect after controlling for the group membership use "within"; if the interest is in the group x time interaction use "interaction"
`verbose`	if `FALSE` no output is printed on the console

Value

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

References

Bulus, M., & Polat, C. (in press). pwrss R paketi ile istatistiksel guc analizi [Statistical power analysis with pwrss R package]. Ahi Evran Universitesi Kirsehir Egitim Fakultesi Dergisi. https://osf.io/ua5fc/download/

Examples

```
#######################################################
# pretest-posttest design with treatment group only  #
#######################################################

# a researcher is expecting a difference of Cohen's d = 0.30
# between posttest and pretest score translating into
# Eta-squared = 0.022
pwrss.f.rmanova(eta2 = 0.022,  n.levels = 1, n.rm = 2,
                corr.rm = 0.50, type = "within",
```

```
                alpha = 0.05, power = 0.80)

# paired t-test approach
pwrss.t.2means(mu1 = 0.30, mu2 = 0,
               sd1 = 1, sd2 = 1,
               paired = TRUE, paired.r = 0.50,
               alpha = 0.05, power = 0.80)

##########################################################
# posttest only design with treatment and control groups #
##########################################################

# a researcher is expecting a difference of Cohen's d = 0.50
# on the posttest score between treatment and control groups
# translating into Eta-squared = 0.059
pwrss.f.rmanova(eta2 = 0.059,  n.levels = 2, n.rm = 1,
                type = "between",
                alpha = 0.05, power = 0.80)

# independent t-test approach
pwrss.t.2means(mu1 = 0.50, mu2 = 0,
               sd1 = 1, sd2 = 1,
               alpha = 0.05, power = 0.80)

#############################################################
# pretest-posttest design with treatment and control groups #
#############################################################

# a researcher is expecting a difference of Cohen's d = 0.40
# on the posttest score between treatment and control groups
# after controlling for the pretest translating into
# partial Eta-squared = 0.038
pwrss.f.rmanova(eta2 = 0.038,  n.levels = 2, n.rm = 2,
                corr.rm = 0.50, type = "between",
                alpha = 0.05, power = 0.80)

# regression approach
p <- 0.50 # proportion of subjects in treatment group
pwrss.t.reg(beta1 = 0.40, r2 = 0.25, k = 2,
            sdx = sqrt(p*(1-p)),
            alpha = 0.05, power = 0.80)

# a researcher is expecting an interaction effect
# (between groups and time) of Eta-squared = 0.01
pwrss.f.rmanova(eta2 = 0.01,  n.levels = 2, n.rm = 2,
                corr.rm = 0.50, type = "interaction",
                alpha = 0.05, power = 0.80)
```

`pwrss.np.2groups` *Difference between Two Groups (Non-parametric Tests for Independent and Paired Samples)*

Description

Calculates statistical power or minimum required sample size (only one can be NULL at a time) to test difference between two groups. Although means and standard deviations are some of the arguments in the function below, what is actually being tested is the difference between mean ranks. First, the mean difference is converted into Cohen's d, and then into probability of superiority, which is the probability of an observation in group 1 being higher than an observation in group 2. Probability of superiority can be extracted as `pwrss.np.2groups()$parms$prob1`. This parameterization, expressed as means and standard deviations, helps in making comparisons and switching back and forth between parametric and non-parametric tests.

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`.

Formulas are validated using Monte Carlo simulation, G*Power, and tables in PASS documentation.

Usage

```
pwrss.np.2groups(mu1 = 0.20, mu2 = 0,
                 sd1 = ifelse(paired, sqrt(1/(2*(1-paired.r))), 1), sd2 = sd1,
                 margin = 0, alpha = 0.05, paired = FALSE, paired.r = 0.50,
                 kappa = 1, n2 = NULL, power = NULL,
                 alternative = c("not equal", "greater", "less",
                                 "non-inferior", "superior", "equivalent"),
                 distribution = c("normal", "uniform", "double exponential",
                                  "laplace", "logistic"),
                 method = c("guenther", "noether"),
                 verbose = TRUE)
```

Arguments

`mu1`	expected mean in the first group
`mu2`	expected mean in the second group
`sd1`	expected standard deviation in the first group
`sd2`	expected standard deviation in the second group
`paired`	if `TRUE` paired samples
`paired.r`	correlation between repeated measures for paired samples (e.g., pretest and posttest)
`n2`	sample size in the second group (or for the single group in paired samples)
`kappa`	`n1/n2` ratio (applies to independent samples only)
`power`	statistical power $(1-\beta)$
`alpha`	probability of type I error
`margin`	non-inferority, superiority, or equivalence margin (margin: boundry of `mu1 - mu2` that is practically insignificant)
`alternative`	direction or type of the hypothesis test: "not equal", "greater", "less", "equivalent", "non-inferior", or "superior"

`distribution`	parent distribution: "normal", "uniform", "double exponential", "laplace", or "logistic"
`method`	non-parametric method: "guenther" (default) or "noether"
`verbose`	if `FALSE` no output is printed on the console

Value

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

References

Al-Sunduqchi, M. S. (1990). Determining the appropriate sample size for inferences based on the Wilcoxon statistics [Unpublished doctoral dissertation]. University of Wyoming - Laramie

Chow, S. C., Shao, J., Wang, H., and Lokhnygina, Y. (2018). Sample size calculations in clinical research (3rd ed.). Taylor & Francis/CRC.

Noether, G. E. (1987). Sample size determination for some common nonparametric tests. Journal of the American Statistical Association, 82(1), 645-647.

Ruscio, J. (2008). A probability-based measure of effect size: Robustness to base rates and other factors. Psychological Methods, 13(1), 19-30.

Ruscio, J., & Mullen, T. (2012). Confidence intervals for the probability of superiority effect size measure and the area under a receiver operating characteristic curve. Multivariate Behavioral Research, 47(2), 201-223.

Zhao, Y.D., Rahardja, D., & Qu, Y. (2008). Sample size calculation for the Wilcoxon-Mann-Whitney test adjusting for ties. Statistics in Medicine, 27(3), 462-468.

Examples

```
# Mann-Whitney U or Wilcoxon rank-sum test
# (a.k.a Wilcoxon-Mann-Whitney test) for independent samples

## difference between group 1 and group 2 is not equal to zero
## estimated difference is Cohen'd = 0.25
pwrss.np.2means(mu1 = 0.25, mu2 = 0, power = 0.80,
                alternative = "not equal")

## difference between group 1 and group 2 is greater than zero
## estimated difference is Cohen'd = 0.25
pwrss.np.2means(mu1 = 0.25, mu2 = 0, power = 0.80,
                alternative = "greater")

## mean of group 1 is practically not smaller than mean of group 2
```

```
## estimated difference is Cohen'd = 0.10 and can be as small as -0.05
pwrss.np.2means(mu1 = 0.25, mu2 = 0.15,
                margin = -0.05, power = 0.80,
                alternative = "non-inferior")

## mean of group 1 is practically greater than mean of group 2
## estimated difference is Cohen'd = 0.10 and can be as small as 0.05
pwrss.np.2means(mu1 = 0.25, mu2 = 0.15,
                margin = 0.05, power = 0.80,
                alternative = "superior")

## mean of group 1 is practically same as mean of group 2
## estimated difference is Cohen'd = 0
## and can be as small as -0.05 and as high as 0.05
pwrss.np.2means(mu1 = 0.25, mu2 = 0.25,
                margin = 0.05, power = 0.80,
                alternative = "equivalent")


# Wilcoxon signed-rank test for matched pairs (dependent samples)

## difference between time 1 and time 2 is not equal to zero
## estimated difference between time 1 and time 2 is Cohen'd = -0.25
pwrss.np.2means(mu1 = 0, mu2 = 0.25, power = 0.80,
                paired = TRUE, paired.r = 0.50,
                alternative = "not equal")

## difference between time 1 and time 2 is greater than zero
## estimated difference between time 1 and time 2 is Cohen'd = -0.25
pwrss.np.2means(mu1 = 0, mu2 = 0.25, power = 0.80,
                paired = TRUE, paired.r = 0.50,
                alternative = "greater")

## mean of time 1 is practically not smaller than mean of time 2
## estimated difference is Cohen'd = -0.10 and can be as small as 0.05
pwrss.np.2means(mu1 = 0.15, mu2 = 0.25, margin = 0.05,
                paired = TRUE, paired.r = 0.50, power = 0.80,
                alternative = "non-inferior")

## mean of time 1 is practically greater than mean of time 2
## estimated difference is Cohen'd = -0.10 and can be as small as -0.05
pwrss.np.2means(mu1 = 0.15, mu2 = 0.25, margin = -0.05,
                paired = TRUE, paired.r = 0.50, power = 0.80,
                alternative = "superior")

## mean of time 1 is practically same as mean of time 2
## estimated difference is Cohen'd = 0
## and can be as small as -0.05 and as high as 0.05
pwrss.np.2means(mu1 = 0.25, mu2 = 0.25, margin = 0.05,
                paired = TRUE, paired.r = 0.50, power = 0.80,
                alternative = "equivalent")
```

pwrss.t.2means *Difference between Two Means (t or z Test for Independent or Paired Samples)*

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`.

Formulas are validated using Monte Carlo simulation, G*Power, http://powerandsamplesize.com/, and tables in PASS documentation.

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)

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`	expected standard deviation in the first group
`sd2`	expected standard deviation in the second group
`paired`	if `TRUE` paired samples t test
`paired.r`	correlation between repeated measures for paired samples (e.g., pretest and posttest)
`n2`	sample size in the second group (or for the single group in paired samples)
`kappa`	`n1/n2` ratio (applies to independent samples only)
`power`	statistical power $(1 - \beta)$
`alpha`	probability of type I error
`welch.df`	if `TRUE` uses Welch's degrees of freedom adjustment when groups sizes or variances are not equal (applies to independent samples t test only)

`margin`	non-inferority, superiority, or equivalence margin (margin: boundry of `mu1 - mu2` that is practically insignificant)
`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

Value

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

References

Bulus, M., & Polat, C. (in press). pwrss R paketi ile istatistiksel guc analizi [Statistical power analysis with pwrss R package]. Ahi Evran Universitesi Kirsehir Egitim Fakultesi Dergisi. https://osf.io/ua5fc/download/

Chow, S. C., Shao, J., Wang, H., & Lokhnygina, Y. (2018). Sample size calculations in clinical research (3rd ed.). Taylor & Francis/CRC.

Cohen, J. (1988). Statistical power analysis for the behavioral sciences (2nd ed.). Lawrence Erlbaum Associates.

Examples

```
# independent samples t test

## difference between group 1 and group 2 is not equal to zero
## estimated difference is Cohen'd = 0.25
pwrss.t.2means(mu1 = 0.25, mu2 = 0, power = 0.80,
                alternative = "not equal")

## difference between group 1 and group 2 is greater than zero
## estimated difference is Cohen'd = 0.25
pwrss.t.2means(mu1 = 0.25, mu2 = 0, power = 0.80,
                alternative = "greater")

## mean of group 1 is practically not smaller than mean of group 2
## estimated difference is Cohen'd = 0.10 and can be as small as -0.05
pwrss.t.2means(mu1 = 0.25, mu2 = 0.15,
                margin = -0.05, power = 0.80,
                alternative = "non-inferior")

## mean of group 1 is practically greater than mean of group 2
## estimated difference is Cohen'd = 0.10 and can be as small as 0.05
pwrss.t.2means(mu1 = 0.25, mu2 = 0.15,
```

```
                 margin = 0.05, power = 0.80,
                 alternative = "superior")

## mean of group 1 is practically same as mean of group 2
## estimated difference is Cohen'd = 0
## and can be as small as -0.05 and as high as 0.05
pwrss.t.2means(mu1 = 0.25, mu2 = 0.25,
                 margin = 0.05, power = 0.80,
                 alternative = "equivalent")


#  dependent samples (matched pairs) t test

## difference between time 1 and time 2 is not equal to zero
## estimated difference between time 1 and time 2 is Cohen'd = -0.25
pwrss.t.2means(mu1 = 0, mu2 = 0.25, power = 0.80,
                 paired = TRUE, paired.r = 0.50,
                 alternative = "not equal")

## difference between time 1 and time 2 is less than zero
## estimated difference between time 1 and time 2 is Cohen'd = -0.25
pwrss.t.2means(mu1 = 0, mu2 = 0.25, power = 0.80,
                 paired = TRUE, paired.r = 0.50,
                 alternative = "less")

## mean of time 1 is practically not smaller than mean of time 2
## estimated difference is Cohen'd = -0.10 and can be as small as 0.05
pwrss.t.2means(mu1 = 0.15, mu2 = 0.25, margin = 0.05,
                 paired = TRUE, paired.r = 0.50, power = 0.80,
                 alternative = "non-inferior")

## mean of time 1 is practically greater than mean of time 2
## estimated difference is Cohen'd = -0.10 and can be as small as -0.05
pwrss.t.2means(mu1 = 0.15, mu2 = 0.25, margin = -0.05,
                 paired = TRUE, paired.r = 0.50, power = 0.80,
                 alternative = "superior")

## mean of time 1 is practically same as mean of time 2
## estimated difference is Cohen'd = 0
## and can be as small as -0.05 and as high as 0.05
pwrss.t.2means(mu1 = 0.25, mu2 = 0.25, margin = 0.05,
                 paired = TRUE, paired.r = 0.50, power = 0.80,
                 alternative = "equivalent")
```

`pwrss.t.mean` *One Mean against a Constant (z or t 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.

Formulas are validated using http://powerandsamplesize.com/, and tables in PASS documentation.

Usage

```
pwrss.t.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)

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 compared (a mean)
`n`	sample size
`power`	statistical power $(1-\beta)$
`alpha`	probability of type I error
`margin`	non-inferority, superiority, or equivalence margin (margin: boundry of `mu - mu0` that is practically insignificant)
`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

Value

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

References

Bulus, M., & Polat, C. (in press). pwrss R paketi ile istatistiksel guc analizi [Statistical power analysis with pwrss R package]. Ahi Evran Universitesi Kirsehir Egitim Fakultesi Dergisi. https://osf.io/ua5fc/download/

Examples

```
# Example: A researcher is expecting a score of 23
# on Beck depression inventory (BDI) which is
# 0.50 standard devation above the threshold value 20
# (assume standard deviation of BDI scores is 6).

# to find that a score of 23 is greater than the threshold 20
pwrss.t.mean(mu = 23, mu0 = 20, sd = 6,
             alpha = 0.05, power = 0.80,
             alternative = "greater")
# standardized formulation
pwrss.t.mean(mu = 0.50, mu0 = 0, sd = 1,
             alpha = 0.05, power = 0.80,
             alternative = "greater")
```

`pwrss.t.reg` *Linear Regression: Single Coefficient (t 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 an experimental design) by specifying `sdx = sqrt(p*(1-p))` where `p` is the proportion of subjects in one of the groups. The sample size in each group would be `n*p` and `n*(1-p)`. `pwrss.t.regression()` and `pwrss.t.reg()` are the same functions.

When HIGHER values of an outcome is a good thing, `beta1` is expected to be greater than `beta0 + margin` for non-inferiority and superiority tests. In this case, `margin` is NEGATIVE for the non-inferiority test but it is POSITIVE for the superiority test.

When LOWER values of an outcome is a good thing, `beta1` is expected to be less than `beta0 + margin` for non-inferiority and superiority tests. In this case, `margin` is POSITIVE for the non-inferiority test but it is NEGATIVE for the superiority test.

For equivalence tests the value of `beta0` shifts both to the left and right as `beta0 - margin` and `beta0 + margin`. For equivalence tests `margin` is stated as the absolute value and `beta1` is expected to fall between `beta0 - margin` and `beta0 + margin`.

Formulas are validated using Monte Carlo simulation, G*Power, tables in PASS documentation, and tables in Bulus (2021).

Usage

```
pwrss.t.reg(beta1 = 0.25, beta0 = 0, margin = 0,
            sdx = 1, sdy = 1,
            k = 1, r2 = (beta1 * sdx / sdy)^2,
            alpha = 0.05, n = NULL, power = NULL,
            alternative = c("not equal", "less", "greater",
```

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

pwrss.z.reg(beta1 = 0.25, beta0 = 0, margin = 0,
            sdx = 1, sdy = 1,
            k = 1, r2 = (beta1 * sdx / sdy)^2,
            alpha = 0.05, n = NULL, power = NULL,
            alternative = c("not equal", "less", "greater",
                         "non-inferior", "superior", "equivalent"),
            verbose = TRUE)
```

Arguments

beta1	expected regression coefficient. One can use standardized regression coefficient, but should keep sdx = 1 and sdy = 1 or leave them out as they are default specifications
beta0	regression coefficient under null hypothesis (usually zero). Not to be confused with the intercept. One can use standardized regression coefficient, but should keep sdx = 1 and sdy = 1 or leave them out as they are default specifications
margin	non-inferiority, superiority, or equivalence margin (margin: boundry of beta1 - beta0 that is practically insignificant)
sdx	expected 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	expected 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 r2 along with the argument k>1.
n	total 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", "non-inferior", "superior", or "equivalent"
verbose	if FALSE no output is printed on the console

Value

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

References

Bulus, M. (2021). Sample size determination and optimal design of randomized/non-equivalent pretest-posttest control-group designs. Adiyaman Univesity Journal of Educational Sciences, 11(1), 48-69.

Phillips, K. F. (1990). Power of the two one-Sided tests procedure in bioequivalence. Journal of Pharmacokinetics and Biopharmaceutics, 18(2), 137-144.

Dupont, W. D., and Plummer, W. D. (1998). Power and sample size calculations for studies involving linear regression. Controlled Clinical Trials, 19(6), 589-601.

Examples

```
# continuous predictor x (and 4 covariates)
pwrss.t.reg(beta1 = 0.20, alpha = 0.05,
            alternative = "not equal",
            k = 5, r2 = 0.30,
            power = 0.80)

pwrss.t.reg(beta1 = 0.20, alpha = 0.05,
            alternative = "not equal",
            k = 5, r2 = 0.30,
            n = 138)

# binary predictor x (and 4 covariates)
p <- 0.50 # proportion of subjects in one group
pwrss.t.reg(beta1 = 0.20, alpha = 0.05,
            alternative = "not equal",
            sdx = sqrt(p*(1-p)),
            k = 5, r2 = 0.30,
            power = 0.80)

pwrss.t.reg(beta1 = 0.20, alpha = 0.05,
            alternative = "not equal",
            sdx = sqrt(p*(1-p)) ,
            k = 5, r2 = 0.30,
            n = 550)

# non-inferiority test with binary predictor x (and 4 covariates)
p <- 0.50 # proportion of subjects in one group
pwrss.t.reg(beta1 = 0.20, beta0 = 0.10, margin = -0.05,
            alpha = 0.05, alternative = "non-inferior",
            sdx = sqrt(p*(1-p)),
            k = 5, r2 = 0.30,
            power = 0.80)

pwrss.t.reg(beta1 = 0.20, beta0 = 0.10, margin = -0.05,
            alpha = 0.05, alternative = "non-inferior",
            sdx = sqrt(p*(1-p)),
            k = 5, r2 = 0.30,
            n = 770)

# superiority test with binary predictor x (and 4 covariates)
```

```
p <- 0.50 # proportion of subjects in one group
pwrss.t.reg(beta1 = 0.20, beta0 = 0.10, margin = 0.01,
            alpha = 0.05, alternative = "superior",
            sdx = sqrt(p*(1-p)),
            k = 5, r2 = 0.30,
            power = 0.80)

pwrss.t.reg(beta1 = 0.20, beta0 = 0.10, margin = 0.01,
            alpha = 0.05, alternative = "superior",
            sdx = sqrt(p*(1-p)),
            k = 5, r2 = 0.30,
            n = 2138)

# equivalence test with binary predictor x (and 4 covariates)
p <- 0.50 # proportion of subjects in one group
pwrss.t.reg(beta1 = 0.20, beta0 = 0.20, margin = 0.05,
            alpha = 0.05, alternative = "equivalent",
            sdx = sqrt(p*(1-p)),
            k = 5, r2 = 0.30,
            power = 0.80)

pwrss.t.reg(beta1 = 0.20, beta0 = 0.20, margin = 0.05,
            alpha = 0.05, alternative = "equivalent",
            sdx = sqrt(p*(1-p)),
            k = 5, r2 = 0.30,
            n = 9592)
```

`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 (Pearson) correlations using Fisher's z transformation.

Formulas are validated using Monte Carlo simulation, G*Power, and tables in PASS documentation.

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. Sample size in the first group can be calculated as `n2*kappa`. By default, `n1 = n2` because `kappa = 1`

kappa	n1/n2 ratio
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

Value

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

References

Bulus, M., & Polat, C. (in press). pwrss R paketi ile istatistiksel guc analizi [Statistical power analysis with pwrss R package]. Ahi Evran Universitesi Kirsehir Egitim Fakultesi Dergisi. https://osf.io/ua5fc/download/

Chow, S. C., Shao, J., Wang, H., & Lokhnygina, Y. (2018). Sample size calculations in clinical research (3rd ed.). Taylor & Francis/CRC.

Cohen, J. (1988). Statistical power analysis for the behavioral sciences (2nd ed.). Lawrence Erlbaum Associates.

Examples

```
# difference between r1 and r2 is different from zero
# it could be -0.10 as well as 0.10
pwrss.z.2corrs(r1 = .20, r2 = 0.30,
               alpha = 0.05, power = .80,
               alternative = "not equal")

# difference between r1 and r2 is greater than zero
pwrss.z.2corrs(r1 = .30, r2 = 0.20,
               alpha = 0.05, power = .80,
               alternative = "greater")
```

---

pwrss.z.2props *Difference between Two Proportions (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.

Formulas are validated using Monte Carlo simulation, G*Power, http://powerandsamplesize.com/ and tables in PASS documentation.

Usage

```
pwrss.z.2props(p1, p2, margin = 0, arcsin.trans = FALSE, 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 arcsine transformation, if `FALSE` uses normal approximation (default)
`kappa`	`n1/n2` ratio
`n2`	sample size in the second group. Sample size in the first group can be calculated as `n2*kappa`. By default, `n1 = n2` because `kappa = 1`
`power`	statistical power $(1-\beta)$
`alpha`	probability of type I error.
`margin`	non-inferority, superiority, or equivalence margin (margin: boundry of `p1 - p2` that is practically insignificant)
`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

Value

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

References

Bulus, M., & Polat, C. (in press). pwrss R paketi ile istatistiksel guc analizi [Statistical power analysis with pwrss R package]. Ahi Evran Universitesi Kirsehir Egitim Fakultesi Dergisi. https://osf.io/ua5fc/download/

Chow, S. C., Shao, J., Wang, H., & Lokhnygina, Y. (2018). Sample size calculations in clinical research (3rd ed.). Taylor & Francis/CRC.

Cohen, J. (1988). Statistical power analysis for the behavioral sciences (2nd ed.). Lawrence Erlbaum Associates.

Examples

```
# Example 1: expecting p1 - p2 smaller than 0
## one-sided test with normal approximation
pwrss.z.2props(p1 = 0.45, p2 = 0.50,
               alpha = 0.05, power = 0.80,
               alternative = "less",
               arcsin.trans = FALSE)
## one-sided test with arcsine transformation
pwrss.z.2props(p1 = 0.45, p2 = 0.50,
               alpha = 0.05, power = 0.80,
               alternative = "less",
               arcsin.trans = TRUE)

# Example 2: expecting p1 - p2 smaller than 0 or greater than 0
## two-sided test with normal approximation
pwrss.z.2props(p1 = 0.45, p2 = 0.50,
               alpha = 0.05, power = 0.80,
               alternative = "not equal",
               arcsin.trans = FALSE)
## two-sided test with arcsine transformation
pwrss.z.2props(p1 = 0.45, p2 = 0.50,
               alpha = 0.05, power = 0.80,
               alternative = "not equal",
               arcsin.trans = TRUE)

# Example 2: expecting p1 - p2 smaller than 0.01
# when smaller proportion is better
## non-inferiority test with normal approximation
pwrss.z.2props(p1 = 0.45, p2 = 0.50, margin = 0.01,
               alpha = 0.05, power = 0.80,
               alternative = "non-inferior",
               arcsin.trans = FALSE)
## non-inferiority test with arcsine transformation
pwrss.z.2props(p1 = 0.45, p2 = 0.50, margin = 0.01,
               alpha = 0.05, power = 0.80,
               alternative = "non-inferior",
               arcsin.trans = TRUE)

# Example 3: expecting p1 - p2 greater than -0.01
# when bigger proportion is better
## non-inferiority test with normal approximation
pwrss.z.2props(p1 = 0.55, p2 = 0.50, margin = -0.01,
               alpha = 0.05, power = 0.80,
               alternative = "non-inferior",
               arcsin.trans = FALSE)
## non-inferiority test with arcsine transformation
pwrss.z.2props(p1 = 0.55, p2 = 0.50,  margin = -0.01,
               alpha = 0.05, power = 0.80,
               alternative = "non-inferior",
               arcsin.trans = TRUE)

# Example 4: expecting p1 - p2 smaller than -0.01
```

```
# when smaller proportion is better
## superiority test with normal approximation
pwrss.z.2props(p1 = 0.45, p2 = 0.50, margin = -0.01,
               alpha = 0.05, power = 0.80,
               alternative = "superior",
               arcsin.trans = FALSE)
## superiority test with arcsine transformation
pwrss.z.2props(p1 = 0.45, p2 = 0.50, margin = -0.01,
               alpha = 0.05, power = 0.80,
               alternative = "superior",
               arcsin.trans = TRUE)

# Example 5: expecting p1 - p2 greater than 0.01
# when bigger proportion is better
## superiority test with normal approximation
pwrss.z.2props(p1 = 0.55, p2 = 0.50, margin = 0.01,
               alpha = 0.05, power = 0.80,
               alternative = "superior",
               arcsin.trans = FALSE)
## superiority test with arcsine transformation
pwrss.z.2props(p1 = 0.55, p2 = 0.50, margin = 0.01,
               alpha = 0.05, power = 0.80,
               alternative = "superior",
               arcsin.trans = TRUE)

# Example 6: expecting p1 - p2 between -0.01 and 0.01
## equivalence test with normal approximation
pwrss.z.2props(p1 = 0.50, p2 = 0.50, margin = 0.01,
               alpha = 0.05, power = 0.80,
               alternative = "equivalent",
               arcsin.trans = FALSE)
# equivalence test with arcsine transformation
pwrss.z.2props(p1 = 0.50, p2 = 0.50, margin = 0.01,
               alpha = 0.05, power = 0.80,
               alternative = "equivalent",
               arcsin.trans = TRUE)
```

`pwrss.z.corr` *One Correlation 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 (Pearson) correlation against a constant using Fisher's z transformation.

Formulas are validated using G*Power and tables in PASS documentation.

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 to be compared (a correlation)
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

Value

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

References

Bulus, M., & Polat, C. (in press). pwrss R paketi ile istatistiksel guc analizi [Statistical power analysis with pwrss R package]. Ahi Evran Universitesi Kirsehir Egitim Fakultesi Dergisi. https://osf.io/ua5fc/download/

Chow, S. C., Shao, J., Wang, H., & Lokhnygina, Y. (2018). Sample size calculations in clinical research (3rd ed.). Taylor & Francis/CRC.

Cohen, J. (1988). Statistical power analysis for the behavioral sciences (2nd ed.). Lawrence Erlbaum Associates.

Examples

```
# expected correlation is 0.20 and it is different from 0
# it could be 0.20 as well as -0.20
pwrss.z.corr(r = 0.20, r0 = 0,
             alpha = 0.05, power = 0.80,
             alternative = "not equal")

# expected correlation is 0.20 and it is greater than 0.10
pwrss.z.corr(r = 0.20, r0 = 0.10,
             alpha = 0.05, power = 0.80,
             alternative = "greater")
```

`pwrss.z.logreg` *Logistic Regression: Single Coefficient (Large Sample Approx. Wald's 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 logistic regression. `pwrss.z.logistic()` and `pwrss.z.logreg()` are the same functions. The distribution of the predictor variable can be one of the following: `c("normal", "poisson", "uniform", "exponential", "binomial", "bernouilli", "lognormal")` for Demidenko (2007) procedure but only `c("normal", "binomial", "bernouilli")` for Hsieh et al. (1998) procedure. The default parameters for these distributions are

```
distribution = list(dist = "normal", mean = 0, sd = 1)
distribution = list(dist = "poisson", lambda = 1)
distribution = list(dist = "uniform", min = 0, max = 1)
distribution = list(dist = "exponential", rate = 1)
distribution = list(dist = "binomial", size = 1, prob = 0.50)
distribution = list(dist = "bernoulli", prob = 0.50)
distribution = list(dist = "lognormal", meanlog = 0, sdlog = 1)
```

Parameters defined in `list()` form can be modified, but the names should be kept the same. It is sufficient to use distribution's name for default parameters (e.g. `dist = "normal"`).

Formulas are validated using Monte Carlo simulation, G*Power, and tables in PASS documentation.

Usage

```
pwrss.z.logreg(p1 = 0.10, p0 = 0.15,
               odds.ratio  = (p1/(1-p1))/(p0/(1-p0)),
               beta0 = log(p0/(1-p0)), beta1 = log(odds.ratio),
               n = NULL, power = NULL, r2.other.x = 0, alpha = 0.05,
               alternative = c("not equal", "less", "greater"),
               method = c("demidenko(vc)", "demidenko", "hsieh"),
               distribution = "normal", verbose = TRUE)


pwrss.z.logistic(p1 = 0.10, p0 = 0.15,
                 odds.ratio  = (p1/(1-p1))/(p0/(1-p0)),
                 beta0 = log(p0/(1-p0)), beta1 = log(odds.ratio),
                 n = NULL, power = NULL, r2.other.x = 0, alpha = 0.05,
                 alternative = c("not equal", "less", "greater"),
                 method = c("demidenko(vc)", "demidenko", "hsieh"),
                 distribution = "normal", verbose = TRUE)
```

Arguments

`p0`	base probability under null hypothesis (probability that an event occurs without the influence of the predictor X - or when the value of the predictor is zero)

`p1`	probability under alternative hypothesis (probability that an event occurs when the value of the predictor X is increased by one unit)
`beta0`	regression coefficient defined as `beta0 = log( p0/(1-p0) )`
`beta1`	regression coefficient for the predictor X defined as `beta1 = log( (p1/(1-p1)) / (p0/(1-p0)) )`
`odds.ratio`	odds ratio defined as `odds.ratio = exp(beta1) = (p1/(1-p1)) / (p0/(1-p0))`
`n`	total sample size
`power`	statistical power $(1 - \beta)$
`r2.other.x`	proportion of variance in the predictor X explained by other covariates. Not to be confused with pseudo R-squared
`alpha`	probability of type I error
`alternative`	direction or type of the hypothesis test: "not equal", "greater", "less"
`method`	calculation method. `"demidenko(vc)"` stands for Demidenko (2007) procedure with variance correction; `"demidenko"` stands for Demidenko (2007) procedure without variance correction; `"hsieh"` stands for Hsieh et al. (1998) procedure. `"demidenko"` and `"hsieh"` methods produce similiar results but `"demidenko(vc)"` is more precise
`distribution`	distribution family. Can be one of the `c("noramal", "poisson", "uniform", "exponential", "binomial", "bernouilli", "lognormal")` for Demidenko (2007) procedure but only `c("normal", "binomial", "bernouilli")` for Hsieh et al. (1998) procedure
`verbose`	if `FALSE` no output is printed on the console

Value

`parms`	list of parameters used in calculation
`test`	type of the statistical test (z test)
`ncp`	non-centrality parameter
`power`	statistical power $(1 - \beta)$
`n`	total sample size

References

Demidenko, E. (2007). Sample size determination for logistic regression revisited. Statistics in Medicine, 26(18), 3385-3397.

Hsieh, F. Y., Bloch, D. A., & Larsen, M. D. (1998). A simple method of sample size calculation for linear and logistic regression. Statistics in Medicine, 17(4), 1623-1634.

Examples

```
# predictor X follows normal distribution

## probability specification
pwrss.z.logreg(p0 = 0.15, p1 = 0.10,
               alpha = 0.05, power = 0.80,
               dist = "normal")

## odds ratio specification
pwrss.z.logreg(p0 = 0.15, odds.ratio = 0.6296,
               alpha = 0.05, power = 0.80,
               dist = "normal")

## regression coefficient specification
pwrss.z.logreg(p0 = 0.15, beta1 = -0.4626,
               alpha = 0.05, power = 0.80,
               dist = "normal")

## change parameters associated with predictor X
dist.x <- list(dist = "normal", mean = 10, sd = 2)
pwrss.z.logreg(p0 = 0.15, beta1 = -0.4626,
               alpha = 0.05, power = 0.80,
               dist = dist.x)


# predictor X follows Bernoulli distribution (such as treatment/control groups)

## probability specification
pwrss.z.logreg(p0 = 0.15, p1 = 0.10,
               alpha = 0.05, power = 0.80,
               dist = "bernoulli")

## odds ratio specification
pwrss.z.logreg(p0 = 0.15, odds.ratio = 0.6296,
               alpha = 0.05, power = 0.80,
               dist = "bernoulli")

## regression coefficient specification
pwrss.z.logreg(p0 = 0.15, beta1 = -0.4626,
               alpha = 0.05, power = 0.80,
               dist = "bernoulli")

## change parameters associated with predictor X
dist.x <- list(dist = "bernoulli", prob = 0.30)
pwrss.z.logreg(p0 = 0.15, beta1 = -0.4626,
               alpha = 0.05, power = 0.80,
               dist = dist.x)
```

`pwrss.z.med` *Indirect Effect in Mediation Analysis (z, Joint, and Monte Carlo Tests)*

Description

Calculates statistical power or minimum required sample size (only one can be NULL at a time) to test indirect effects in mediation analysis (z test, joint test, and Monte Carlo test). One can consider explanatory power of the covariates in the mediator and outcome model via specifying R-squared values accordingly. `pwrss.z.mediation()` and `pwrss.z.med()` are the same functions.

Formulas are validated using Monte Carlo simulation.

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 keep `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 keep `sdm = 1` and `sdy = 1` or leave them out as they are default specifications
`cp`	expected regression coefficient for X -> Y path (the direct path). One can use standardized regression coefficient, but should keep `sdx = 1` and `sdy = 1` or leave them out as they are default specifications
`sdx`	expected 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`	expected standard deviation of the mediator (M)
`sdy`	expected 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 mediator 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 outcome model provide a value greater than the default
`n`	total sample size
`power`	statistical power $(1-\beta)$
`alpha`	probability of type I error

alternative	direction of the hypothesis test: "not equal", "greater", "less". It applies to all tests (for path 'a', 'b', and the indirect effect) and typically specified as "not equal". If path 'a' and 'b' have the opposite signs there will be a warning for "greater" or "less" tests (it can be ignored)
mc	logical; if TRUE, statistical power is based on monte carlo simulation
nsims	number of replications (applies when mc = TRUE)
ndraws	number of draws from the distribution of the path coefficients for each replication (applies when mc = TRUE)
verbose	if FALSE no output is printed on the console

Value

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

References

Aroian, L. A. (1947). The probability function of the product of two normally distributed variables. Annals of Mathematical Statistics, 18(2), 265-271.

Goodman, L. A. (1960). On the exact variance of products. Journal of the American Statistical Association, 55(292), 708-713.

MacKinnon, D. P., & Dwyer, J. H. (1993). Estimating mediated effects in prevention studies. Evaluation Review, 17(2), 144-158.

MacKinnon, D. P., Warsi, G., & Dwyer, J. H. (1995). A simulation study of mediated effect measures. Multivariate Behavioral Research, 30(1), 41-62.

Preacher, K. J., & Hayes, A. F. (2004). SPSS and SAS procedures for estimating indirect effects in simple mediation models. Behavior Research Methods, Instruments, & Computers, 36, 717-731.

Preacher, K. J., & Hayes, A. F. (2008). Asymptotic and resampling strategies for assessing and comparing indirect effects in multiple mediator models. Behavior Research Methods, 40, 879-891.

Sobel, M. E. (1982). Asymptotic intervals for indirect effects in structural equations models. In S. Leinhart (Ed.), Sociological methodology 1982 (pp. 290-312). Jossey-Bass.

Examples

```
# with standardized coefficients

## statistical power
pwrss.z.med(a = 0.25, b = 0.25, cp = 0.10,
            alpha = 0.05, n = 200, mc = TRUE)

## minimum required sample size
pwrss.z.med(a = 0.25, b = 0.25, cp = 0.10,
```

```
            alpha = 0.05, power = 0.80)

## adjust for covariates in the outcome model
pwrss.z.med(a = 0.25, b = 0.25, cp = 0.10,
            r2y.mx = 0.50,
            alpha = 0.05, 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

## statistical power
p <- 0.50 # proportion of subjects in one group
pwrss.z.med(a = 0.40, b = 0.25, cp = 0.10,
            sdx = sqrt(p*(1-p)),
            alpha = 0.05, n = 200, mc = TRUE)

## minimum required sample size
pwrss.z.med(a = 0.40, b = 0.25, cp = 0.10,
            sdx = sqrt(p*(1-p)),
            alpha = 0.05, power = 0.80)

## adjust for covariates in outcome model
pwrss.z.med(a = 0.40, b = 0.25, cp = 0.10,
            r2y.mx = 0.50, sdx = sqrt(p*(1-p)),
            alpha = 0.05, power = 0.80)
```

`pwrss.z.poisreg` *Poisson Regression: Single Coefficient (Large Sample Approx. Wald's 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 poisson regression. `pwrss.z.poisson()` and `pwrss.z.poisreg()` are the same functions. The distribution of the predictor variable can be one of the following: `c("normal", "poisson", "uniform", "exponential", "binomial", "bernouilli", "lognormal")`. The default parameters for these distributions are

`distribution = list(dist = "normal", mean = 0, sd = 1)`
`distribution = list(dist = "poisson", lambda = 1)`
`distribution = list(dist = "uniform", min = 0, max = 1)`
`distribution = list(dist = "exponential", rate = 1)`
`distribution = list(dist = "binomial", size = 1, prob = 0.50)`
`distribution = list(dist = "bernoulli", prob = 0.50)`
`distribution = list(dist = "lognormal", meanlog = 0, sdlog = 1)`

Parameters defined in `list()` form can be modified, but the names should be kept the same. It is sufficient to use distribution's name for default parameters (e.g. `dist = "normal"`).

Formulas are validated using Monte Carlo simulation, G*Power, and tables in PASS documentation.

Usage

```
pwrss.z.poisreg(exp.beta0 = 1.10, exp.beta1 = 1.16,
                beta0 = log(exp.beta0), beta1 = log(exp.beta1),
                mean.exposure = 1, n = NULL, power = NULL, r2.other.x = 0,
                alpha = 0.05, alternative = c("not equal", "less", "greater"),
                method = c("demidenko(vc)", "demidenko", "signorini"),
                distribution = "normal", verbose = TRUE)


pwrss.z.poisson(exp.beta0 = 1.10, exp.beta1 = 1.16,
                beta0 = log(exp.beta0), beta1 = log(exp.beta1),
                mean.exposure = 1, n = NULL, power = NULL, r2.other.x = 0,
                alpha = 0.05, alternative = c("not equal", "less", "greater"),
                method = c("demidenko(vc)", "demidenko", "signorini"),
                distribution = "normal", verbose = TRUE)
```

Arguments

`exp.beta0`	the base mean event rate
`exp.beta1`	event rate ratio: the relative increase in the mean event rate for one unit increase in the predictor X (similiar to odds ratio in logistic regression)
`beta0`	`log(exp.beta0)` or natural logarithm of the base mean event rate
`beta1`	`log(exp.beta1)` or natural logarithm of the relative increase in the mean event rate for one unit increase in the predictor X
`mean.exposure`	the mean exposure time (should be > 0). Usually 1
`n`	total sample size
`power`	statistical power $(1-\beta)$
`r2.other.x`	proportion of variance in the predictor X explained by other covariates. Not to be confused with the pseudo R-squared
`alpha`	probability of type I error
`alternative`	direction or type of the hypothesis test: "not equal", "greater", "less"
`method`	calculation method. `"demidenko(vc)"` stands for Demidenko (2007) procedure with variance correction; `"demidenko"` stands for Demidenko (2007) procedure without variance correction; `"signorini"` stands for Signorini (1991) procedure. `"demidenko"` and `"signorini"` methods produce similiar results but `"demidenko(vc)"` is more precise
`distribution`	distribution family. Can be one of the `c("normal", "poisson", "uniform", "exponential", "binomial", "bernouilli", "lognormal")`
`verbose`	if `FALSE` no output is printed on the console

Value

`parms`	list of parameters used in calculation
`test`	type of the statistical test (z test)

ncp	non-centrality parameter
power	statistical power $(1-\beta)$
n	total sample size

References

Demidenko, E. (2007). Sample size determination for logistic regression revisited. Statistics in Medicine, 26(18), 3385-3397.

Hsieh, F. Y., Bloch, D. A., & Larsen, M. D. (1998). A simple method of sample size calculation for linear and logistic regression. Statistics in Medicine, 17(4), 1623-1634.

Signorini, D. F. (1991). Sample size for poisson regression. Biometrika, 78(2), 446-450.

Examples

```
# predictor X follows normal distribution

## regression coefficient specification
pwrss.z.poisreg(beta0 = 0.50, beta1 = -0.10,
                alpha = 0.05, power = 0.80,
                dist = "normal")

## rate ratio specification
pwrss.z.poisreg(exp.beta0 = exp(0.50),
                exp.beta1 = exp(-0.10),
                alpha = 0.05, power = 0.80,
                dist = "normal")

## change parameters associated with predictor X
dist.x <- list(dist = "normal", mean = 10, sd = 2)
pwrss.z.poisreg(exp.beta0 = exp(0.50),
                exp.beta1 = exp(-0.10),
                alpha = 0.05, power = 0.80,
                dist = dist.x)


# predictor X follows Bernoulli distribution (such as treatment/control groups)

## regression coefficient specification
pwrss.z.poisreg(beta0 = 0.50, beta1 = -0.10,
                alpha = 0.05, power = 0.80,
                dist = "bernoulli")

## rate ratio specification
pwrss.z.poisreg(exp.beta0 = exp(0.50),
                exp.beta1 = exp(-0.10),
                alpha = 0.05, power = 0.80,
                dist = "bernoulli")

## change parameters associatied with predictor X
dist.x <- list(dist = "bernoulli", prob = 0.30)
pwrss.z.poisreg(exp.beta0 = exp(0.50),
```

```
exp.beta1 = exp(-0.10),
alpha = 0.05, power = 0.80,
dist = dist.x)
```

`pwrss.z.prop` *One Proportion against a Constant (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.

Formulas are validated using Monte Carlo simulation, G*Power, http://powerandsamplesize.com/ and tables in PASS documentation.

Usage

```
pwrss.z.prop(p, p0 = 0, margin = 0, arcsin.trans = FALSE, 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 compared (a proportion)
`arcsin.trans`	if `TRUE` uses Cohen's arcsine transformation, if `FALSE` uses normal approximation (default)
`n`	sample size
`power`	statistical power $(1 - \beta)$
`alpha`	probability of type I error.
`margin`	non-inferority, superiority, or equivalence margin (margin: boundry of `p - p0` that is practically insignificant)
`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

Value

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

References

Bulus, M., & Polat, C. (in press). pwrss R paketi ile istatistiksel guc analizi [Statistical power analysis with pwrss R package]. Ahi Evran Universitesi Kirsehir Egitim Fakultesi Dergisi. https://osf.io/ua5fc/download/

Chow, S. C., Shao, J., Wang, H., & Lokhnygina, Y. (2018). Sample size calculations in clinical research (3rd ed.). Taylor & Francis/CRC.

Cohen, J. (1988). Statistical power analysis for the behavioral sciences (2nd ed.). Lawrence Erlbaum Associates.

Examples

```
# Example 1: expecting p - p0 smaller than 0
## one-sided test with normal approximation
pwrss.z.prop(p = 0.45, p0 = 0.50,
             alpha = 0.05, power = 0.80,
             alternative = "less",
             arcsin.trans = FALSE)
## one-sided test with arcsine transformation
pwrss.z.prop(p = 0.45, p0 = 0.50,
             alpha = 0.05, power = 0.80,
             alternative = "less",
             arcsin.trans = TRUE)

# Example 2: expecting p - p0 smaller than 0 or greater than 0
## two-sided test with normal approximation
pwrss.z.prop(p = 0.45, p0 = 0.50,
             alpha = 0.05, power = 0.80,
             alternative = "not equal",
             arcsin.trans = FALSE)
## two-sided test with arcsine transformation
pwrss.z.prop(p = 0.45, p0 = 0.50,
             alpha = 0.05, power = 0.80,
             alternative = "not equal",
             arcsin.trans = TRUE)

# Example 2: expecting p - p0 smaller than 0.01
# when smaller proportion is better
## non-inferiority test with normal approximation
pwrss.z.prop(p = 0.45, p0 = 0.50, margin = 0.01,
             alpha = 0.05, power = 0.80,
             alternative = "non-inferior",
             arcsin.trans = FALSE)
## non-inferiority test with arcsine transformation
pwrss.z.prop(p = 0.45, p0 = 0.50, margin = 0.01,
             alpha = 0.05, power = 0.80,
             alternative = "non-inferior",
             arcsin.trans = TRUE)

# Example 3: expecting p - p0 greater than -0.01
# when bigger proportion is better
## non-inferiority test with normal approximation
```

```
pwrss.z.prop(p = 0.55, p0 = 0.50, margin = -0.01,
             alpha = 0.05, power = 0.80,
             alternative = "non-inferior",
             arcsin.trans = FALSE)
## non-inferiority test with arcsine transformation
pwrss.z.prop(p = 0.55, p0 = 0.50, margin = -0.01,
             alpha = 0.05, power = 0.80,
             alternative = "non-inferior",
             arcsin.trans = TRUE)

# Example 4: expecting p - p0 smaller than -0.01
# when smaller proportion is better
## superiority test with normal approximation
pwrss.z.prop(p = 0.45, p0 = 0.50, margin = -0.01,
             alpha = 0.05, power = 0.80,
             alternative = "superior",
             arcsin.trans = FALSE)
## superiority test with arcsine transformation
pwrss.z.prop(p = 0.45, p0 = 0.50, margin = -0.01,
             alpha = 0.05, power = 0.80,
             alternative = "superior",
             arcsin.trans = TRUE)

# Example 5: expecting p - p0 greater than 0.01
# when bigger proportion is better
## superiority test with normal approximation
pwrss.z.prop(p = 0.55, p0 = 0.50, margin = 0.01,
             alpha = 0.05, power = 0.80,
             alternative = "superior",
             arcsin.trans = FALSE)
## superiority test with arcsine transformation
pwrss.z.prop(p = 0.55, p0 = 0.50, margin = 0.01,
             alpha = 0.05, power = 0.80,
             alternative = "superior",
             arcsin.trans = TRUE)

# Example 6: expecting p - p0 between -0.01 and 0.01
## equivalence test with normal approximation
pwrss.z.prop(p = 0.50, p0 = 0.50, margin = 0.01,
             alpha = 0.05, power = 0.80,
             alternative = "equivalent",
             arcsin.trans = FALSE)
# equivalence test with arcsine transformation
pwrss.z.prop(p = 0.50, p0 = 0.50, margin = 0.01,
             alpha = 0.05, power = 0.80,
             alternative = "equivalent",
             arcsin.trans = TRUE)
```

Index