Package 'Sample.Size'

February 19, 2015

Type Package

Title Sample size calculation

Version 1.0

Date 2013-12-03

Author Wei Jiang, Jonathan Mahnken, Matthew Mayo

Maintainer Wei Jiang<wjiang@kumc.edu>

Description Computes the required sample size using the optimal designs with multiple constraints proposed in Mayo et al.(2010). This optimal method is designed for two-arm, randomized phase II clinical trials, and the required sample size can be optimized either using fixed or flexible randomization allocation ratios.

License GPL-2

NeedsCompilation no

Repository CRAN

Date/Publication 2013-12-05 18:05:23

R topics documented:

`SampleSize_OptimalDesign-package`
Sample size calculation

Description

Computes the required sample size using the optimal designs with multiple constraints proposed in Mayo et al.(2010). This optimal method is designed for two-arm, randomized phase II clinical trials, and the required sample size can be optimized either using fixed or flexible randomization allocation ratios.

Details

Package:	SampleSize_OptimalDesign
Type:	Package
Version:	1.0
Date:	2013-12-03
License:	GPL-2

Author(s)

Wei Jiang, Jonathan Mahnken, Matthew Mayo Maintainer: Wei Jiang<wjiang@kumc.edu>

Examples

```
Sample.Size(0.3, 0.6, 0.15, 0.15, 0.15, Allratio_c = 1, Allratio_e = 3)
Sample.Size(0.3, 0.6, 0.15, 0.15, 0.15)
```

`Sample.Size` *Sample size calculation*

Description

Computes the required sample size using the optimal designs with multiple constraints proposed in Mayo et al.(2010). This optimal method is designed for two-arm, randomized phase II clinical trials, and the required sample size can be optimized either using fixed or flexible randomization allocation ratios.

Usage

```
Sample.Size(pi_c, pi_e, gamma_c, gamma_e, gamma_delta, Allratio_c = NA, Allratio_e = NA)
```

Arguments

`pi_c`	Response rate for control
`pi_e`	Response rate for experiment
`gamma_c`	Upper bound for gammaC
`gamma_e`	Upper bound for gammaE
`gamma_delta`	Upper bound for gammaDelta
`Allratio_c`	Allocation Design
`Allratio_e`	Allocation Design

Value

Prints required sample sizes, 1 to 1 allocation design and 1 to 3 allocation design

Author(s)

Wei Jiang, Jonathan Mahnken, Matthew Mayo Maintainer: Wei Jiang<wjiang@kumc.edu>

Examples

```
Sample.Size(0.3, 0.6, 0.15, 0.15, 0.15, Allratio_c = 1, Allratio_e = 3)
Sample.Size(0.3, 0.6, 0.15, 0.15, 0.15)
## The function is currently defined as
function (pi_c, pi_e, gamma_c, gamma_e, gamma_delta, Allratio_c = NA,
    Allratio_e = NA)
{
    c1 <- 1
    e1 <- 1
    dmax1 <- ceiling(max(((pi_c * (1 - pi_c))/(c1 * gamma_c^2)),
        ((pi_e * (1 - pi_e))/(e1 * gamma_e^2)), ((pi_c * (1 -
            pi_c) + (c1/e1) * pi_e * (1 - pi_e))/(c1 * gamma_delta^2))))
    nc_fixed1 <- c1 * dmax1
    ne_fixed1 <- e1 * dmax1
    n_fixed1 <- nc_fixed1 + ne_fixed1
    nc_ast <- (pi_c * (1 - pi_c) + sqrt(pi_c * (1 - pi_c) * pi_e *
        (1 - pi_e)))/gamma_delta^2
    ne_ast <- pi_e * (1 - pi_e) * (gamma_delta^2 - pi_c * (1 -
        pi_c) * nc_ast^(-1))^(-1)
    nc_prime <- pi_c * (1 - pi_c)/gamma_c^2
    fnc_prime <- pi_e * (1 - pi_e) * (gamma_delta^2 - pi_c *
        (1 - pi_c) * nc_prime^(-1))^(-1)
    ne_prime <- pi_e * (1 - pi_e)/gamma_e^2
    fne_prime <- pi_c * (1 - pi_c) * (gamma_delta^2 - pi_e *
        (1 - pi_e) * ne_prime^(-1))^(-1)
    if (nc_ast > nc_prime & ne_ast > ne_prime) {
        c1 <- ceiling(nc_ast)
        e1 <- floor(ne_ast)
        c2 <- floor(nc_ast)
        e2 <- ceiling(ne_ast)
        dmax1 <- max(((pi_c * (1 - pi_c))/(c1 * gamma_c^2)),
            ((pi_e * (1 - pi_e))/(e1 * gamma_e^2)), ((pi_c *
                (1 - pi_c) + (c1/e1) * pi_e * (1 - pi_e))/(c1 *
                gamma_delta^2)))
        dmax2 <- max(((pi_c * (1 - pi_c))/(c2 * gamma_c^2)),
            ((pi_e * (1 - pi_e))/(e2 * gamma_e^2)), ((pi_c *
                (1 - pi_c) + (c2/e2) * pi_e * (1 - pi_e))/(c2 *
                gamma_delta^2)))
        if (dmax1 <= 1) {
            nc_flex <- ceiling(nc_ast)
            ne_flex <- floor(ne_ast)
            n_flex <- nc_flex + ne_flex
            sample_flex <- data.frame(nc_flex, ne_flex, n_flex)
            sample_flex <- data.frame(nc_flex, ne_flex, n_flex)
```

```
    }
    else if (dmax2 <= 1) {
        nc_flex <- floor(nc_ast)
        ne_flex <- ceiling(ne_ast)
        n_flex <- nc_flex + ne_flex
        sample_flex <- data.frame(nc_flex, ne_flex, n_flex)
        sample_flex <- data.frame(nc_flex, ne_flex, n_flex)
    }
    else {
        nc_flex <- ceiling(nc_ast)
        ne_flex <- ceiling(ne_ast)
        n_flex <- nc_flex + ne_flex
        sample_flex <- data.frame(nc_flex, ne_flex, n_flex)
    }
}
else if (nc_ast <= nc_prime & ne_ast > ne_prime) {
    c1 <- ceiling(nc_prime)
    e1 <- floor(max(fnc_prime, ne_prime))
    dmax1 <- max(((pi_c * (1 - pi_c))/(c1 * gamma_c^2)),
        ((pi_e * (1 - pi_e))/(e1 * gamma_e^2)), ((pi_c *
            (1 - pi_c) + (c1/e1) * pi_e * (1 - pi_e))/(c1 *
            gamma_delta^2)))
    if (dmax1 <= 1) {
        nc_flex <- ceiling(nc_prime)
        ne_flex <- floor(max(fnc_prime, ne_prime))
        n_flex <- nc_flex + ne_flex
        sample_flex <- data.frame(nc_flex, ne_flex, n_flex)
    }
    else {
        nc_flex <- ceiling(nc_prime)
        ne_flex <- ceiling(max(fnc_prime, ne_prime))
        n_flex <- nc_flex + ne_flex
        sample_flex <- data.frame(nc_flex, ne_flex, n_flex)
    }
}
else if (nc_ast > nc_prime & ne_ast <= ne_prime) {
    c1 <- floor(max(fne_prime, nc_prime))
    e1 <- ceiling(ne_prime)
    dmax1 <- max(((pi_c * (1 - pi_c))/(c1 * gamma_c^2)),
        ((pi_e * (1 - pi_e))/(e1 * gamma_e^2)), ((pi_c *
            (1 - pi_c) + (c1/e1) * pi_e * (1 - pi_e))/(c1 *
            gamma_delta^2)))
    if (dmax1 <= 1) {
        nc_flex <- floor(max(fne_prime, nc_prime))
        ne_flex <- ceiling(ne_prime)
        n_flex <- nc_flex + ne_flex
        sample_flex <- data.frame(nc_flex, ne_flex, n_flex)
    }
    else {
        nc_flex <- ceiling(max(fne_prime, nc_prime))
        ne_flex <- ceiling(ne_prime)
        n_flex <- nc_flex + ne_flex
        sample_flex <- data.frame(nc_flex, ne_flex, n_flex)
```

```
        }
    }
    else if (nc_ast <= nc_prime & ne_ast <= ne_prime) {
        nc_flex <- ceiling(nc_prime)
        ne_flex <- ceiling(ne_prime)
        n_flex <- nc_flex + ne_flex
        sample_flex <- data.frame(nc_flex, ne_flex, n_flex)
    }
    cat("Specified values for parameters:", "\n")
    cat("Response rates:", "\n", paste("control =", pi_c, "experiment =",
        pi_e), "\n")
    cat("Upper bounds for constriants:", "\n", "gammaC =", gamma_c,
        "gammaE =", gamma_e, "gammaDelta =", gamma_delta, "\n",
        "\n", "\n")
    cat("Required sample sizes:", "\n")
    cat("[1] Optimal Design:", "\n", paste("nc =", nc_flex, "ne =",
        ne_flex, "n =", n_flex), "\n")
    cat("[2] 1 to 1 Allocation Design:", "\n", paste("nc =",
        nc_fixed1, "ne =", ne_fixed1, "n =", n_fixed1), "\n")
    if (!is.na(Allratio_c) | !is.na(Allratio_e)) {
        c <- Allratio_c
        e <- Allratio_e
        dmax <- ceiling(max(((pi_c * (1 - pi_c))/(c * gamma_c^2)),
            ((pi_e * (1 - pi_e))/(e * gamma_e^2)), ((pi_c * (1 -
                pi_c) + (c/e) * pi_e * (1 - pi_e))/(c * gamma_delta^2))))
        nc_fixed <- c * dmax
        ne_fixed <- e * dmax
        n_fixed <- nc_fixed + ne_fixed
        sample_fixed2 <- data.frame(nc_fixed, ne_fixed, n_fixed)
        cat(paste("[3]", c, "to", e, "Allocation Design:"), "\n",
            paste("nc =", nc_fixed, "ne =", ne_fixed, "n =",
                n_fixed), "\n")
    }
}
```

Index