OPERATING-CHARACTERISTIC GUIDED DESIGN OF GROUP-SEQUENTIAL TRIALS

ANIKO SZABO

ABSTRACT. Group-sequential designs are commonly used for clinical trials to allow early stopping for efficacy or futility. While the design of a single-stage randomized trial is guided by a target power for an alternative hypothesis of interest, the addition of interim analyses is driven by technical choices that are less understandable for clinicians. For example, the commonly used Lan-DeMets methodology requires specification of the timing of analyses and error spending functions. Since the rationale and effect of these technical choices is often unclear, the operating characteristics of the final design are explored under various values of the parameter of interest, and the design is then adjusted until desired properties are obtained.

In this work we develop methods for constructing designs that achieve the desired operating characteristics without the need to specify error spending functions or the timing of analyses. Specifically, we consider designing a study for the mean difference θ of a normally distributed outcome with known variance. The null hypothesis $H_0 : \theta = \theta_0$ is tested versus $H_a : \theta = \theta_A$, with power π at a significance level α. The interim analyses are designed so that for a pre-specified sequence θ_{Ak} the study stops for efficacy at stage k with probability π if $\theta = \theta_{Ak}$. If stopping for futility is also considered, then the requirement to stop for futility at stage k with probability π_F if $\theta = \theta_{0k}$ for pre-specified sequence θ_{0k} can also be added. We show that under some monotonicity restrictions, such designs exist for any choice of the timing of interim analyses. Specific designs can be selected by imposing additional optimality requirements, such as minimizing the expected sample size under the target alternative θ_A, or the average sample size under a weighted selection of the alternatives.

1. SETUP AND NOTATIONS

All the calculations are set up assuming a one-sample setting with a normally distributed outcome with a known variance, but many common settings can be reduced to / approximated by this special case.

Let $X_i \sim N(\theta, 1)$ be a sequence of independent identically distributed measurements. We are designing a study to test the following hypotheses:

$$H_0 : \theta = \theta_0 \quad \text{versus} \quad H_A : \theta \geq \theta_0 \tag{1}$$

with type I error α and power π at a pre-specified value $\theta = \theta_A$.

The test statistic for H_0 based on n values is $Z(n) = (\bar{X}_n - \theta_0)\sqrt{n}$, and in general

$$Z(n) \sim N\big(\sqrt{n}(\theta - \theta_0), 1\big). \tag{2}$$

The study will use a group-sequential design with K analyses, conducted at relative times $t_1, \ldots, t_K = 1$. The k^{th} analysis will be conducted after $n_k = N \times t_k$ observations have been obtained, where N is the maximum sample size of the study. Based on the test statistic $Z_k = Z(n_k)$ at analysis $k = 1, \ldots, K-1$, the trial can be stopped early for efficacy (reject H_0) if $Z_k \geq u_k$, stopped early for futility (fail to reject H_0) if $Z_k \leq l_k$, or continued to the next stage. At the final, K^{th} analysis, H_0 would either be rejected if $Z_K \geq u_K = l_K$, or we would fail to reject H_0 otherwise. As a special case, setting $l_k = -\infty$ will result in a study when interim stopping for futility is not considered.

The decision boundaries $u_k, l_k, k = 1, \ldots, K$, and the maximum sample size N have to be selected so that the overall type I error and power of the study are maintained at their design values, but there are many valid choices. In this *operating characteristic driven* approach we propose to select the boundaries based on the cumulative probabilities of stopping for efficacy or futility at each stage.

Date: September 10, 2020.

Table 1. Target operating characteristics

Characteristic	Event	θ	Probability goal
Overall type I error	$\bigcup_{i=1}^{K}\{l_1 < Z_1 < u_1, \ldots, l_{i-1} < Z_{i-1} < u_{i-1}, Z_K \geq u_K\}$	θ_0	$\leq \alpha$
Overall power	$\bigcup_{i=1}^{K}\{l_1 < Z_1 < u_1, \ldots, l_{i-1} < Z_{i-1} < u_{i-1}, Z_K \geq u_K\}$	θ_A	$\geq \pi$
Stop by stage k for efficacy	$\bigcup_{i=1}^{k}\{l_1 < Z_1 < u_1, \ldots, l_{i-1} < Z_{i-1} < u_{i-1}, Z_k \geq u_k\}$	θ_{Ak}	$\geq \pi$
Stop by stage k for futility	$\bigcup_{i=1}^{k}\{l_1 < Z_1 < u_1, \ldots, l_{i-1} < Z_{i-1} < u_{i-1}, Z_k \leq l_k\}$	θ_{0k}	$\geq \pi_F$

2. Main function: calculate design

"../R/gsDesignOC.R" ?≡

```
#'Find optimal group-sequential design
#'
#'The \code{gsDesignOC} function finds the analysis times, boundaries, and sample size
#'for a group-sequential trial
#'
#'@export
#'@param thA numeric; effect size under the alternative hypothesis
#'@param thA.seq monotone numeric vector of values getting closer to thA from outside the
#'th0-thA range (ie, if thA > th0, they should form a decreasing sequence with all values > thA). #'For t
#'@param th0 numeric; effect size under the null hypothesis
#'@param th0.seq monotone numeric vector of values getting closer to th0 from outside the
#'th0-thA range (ie, if thA > th0, they should form an increasing sequence with all values < th0).
#'The study will stop for futility at or before the k-th stage with probability \code{power.futility}.
#'Its length should be equal to that of \code{thA.seq}. The default value of \code{NULL} implies no
#'futility monitoring.
#'@param min.under character string, one of \code{c("alt","null","alt.mixture", "null.mixture")}
#'or their unique abbreviations. Specifies the hypothesis under which the study sample size is
#'minimized. \code{min.under="alt"} minimizes under the alternative hypothesis (theta=\code{thA}),
#'\code{min.under="null"} under the null hypothesis (theta=\code{th0}), while the \code{"mixture"}
#'option minimize the weighted average of sample sizes under theta=\code{thA.seq} or theta=\code{th0.seq}
#'The weights are specified in \code{mix.w}.
#'@param sig.level numeric; the one-sided significance level. The default is 0.025.
#'@param sig.level_final numeric; the desired nominal significance level for testing the null
#'hypothesis at the final stage. Should be between 0 and \code{sig.level}. If NULL, the value
#'will be found using the optimization criterion.
#'@param power numeric; the desired study power. The default is 0.9. This value will also be
#'used to set the probability of stopping for efficacy at stage k under \code{thA.seq}.
#'@param power.futility numeric; the probability of stopping for futility at stage k under \code{th0.seq}
#'@param futility.type character string, one of \code{c("none", "non-binding","binding")} or their
#'unique abbreviations. Specifies whether a futility boundary should not be used at all ("none"), or if i
#'is used, then whether the effect of the futility boundary should be included
#'in the efficacy power/type I error calculations ("binding"), or not ("non-binding").
#'@param mix.w numeric vector of length equal to that of \code{thA.seq}. The weights of the
#'elements of \code{thA.seq} or \code{th0.seq} in the optimality criterion when using \code{min.under="al
#'or \code{min.under="null.mixture"}, respectively. It will be normalized to a sum of 1.
#'Defaults to equal weights.
#'@param control list of parameters controlling the estimation altorithm to replace the default
#'values returned by the \code{glsControl} function.
#'@return an object of class \code{gsDesignOC}
#'@author Aniko Szabo
#'@references Szabo, A, Tarima, S (?) Operating-characteristic guided design of group-sequential trials.
#'@keywords nonparametric design
#'@examples
#'
#'gsDesignOC(0.3, thA.seq = c(1, 0.5), min.under="alt")
#'
#'@name gsDesignOC

gsDesignOC <- function(thA, thA.seq, th0=0, th0.seq=NULL,
                       min.under=c("alt","null","alt.mixture", "null.mixture"),
                       sig.level = 0.025, sig.level_final=NULL,
                       power=0.9, power.futility = power,
                       futility.type=c("none","non-binding","binding"),
                       mix.w = rep(1, length(thA.seq)),
                       control=list()){
  ⟨ Check inputs ? ⟩

  # create skeleton gsDesignOC object
  res <- list(thA=thA, thA.seq=thA.seq,
              th0=th0, th0.seq=th0.seq,
              sig.level = sig.level, sig.level_final=sig.level_final,
              power=power, power.futility = power.futility, futility.type=futility.type)
  class(res) <- "gsDesignOC"
```

⟨ *Check inputs ?* ⟩ ≡

```
if (any(diff(c(thA.seq, thA)) * sign(th0-thA) <= 0))
  stop("'thA.seq' should approach thA in a monotone sequence outside the th0-thA range")

if (!is.null(th0.seq)){
  if (length(th0.seq) != length(thA.seq))
    stop("If specified, 'th0.seq' should have the same length as 'thA.seq'")
  if (any(diff(c(th0.seq, th0)) * sign(thA-th0) <= 0))
    stop("'th0.seq' should approach th0 in a monotone sequence outside the th0-thA range")
}

min.under <- match.arg(min.under)
if (min.under == "alt"){
  mix.theta <- thA
  mix.w <- 1
}
else if (min.under == "null"){
  mix.theta <- th0
  mix.w <- 1
}
else if (min.under == "alt.mixture"){
  if (length(mix.w) != length(thA.seq))
    stop("'mix.w' should have the same length as 'thA.seq' when optimizing under the mixture of alterna
  mix.theta <- thA.seq
}
else if (min.under == "null.mixture"){
  if (is.null(th0.seq))
    stop("'th0.seq' has to be defined when optimizing under the mixture of nulls")
  if (length(mix.w) != length(th0.seq))
    stop("'mix.w' should have the same length as 'th0.seq' when optimizing under the mixture of nulls")
  mix.theta <- th0.seq
}

if (!is.null(sig.level_final) && ((sig.level_final <= 0 || sig.level_final > sig.level)))
  stop("'sig.level_final' should be between 0 and 'sig.level'")

futility.type <- match.arg(futility.type)
if (futility.type != "none"){
  if (is.na(th0.seq)) stop("`th0.seq` should be specified if a futility bound is requested")
}

controlvals <- ocControl()
if (!missing(control))
  controlvals[names(control)] <- control
```

◇

Fragment referenced in ?.

⟨ *Define optimization function ?* ⟩ ≡

```
# n.vec is stage-specific sample sizes
.cp <- function(n.vec){

  n <- sum(n.vec)
  timing <- cumsum(n.vec/n)
  if (!is.null(sig.level_final)) timing <- head(timing, -1)

  res$n <- n
  res$timing <- timing
  res$bounds <- calc.bounds(res)
  dp <- oc(res, EN_theta=mix.theta, mix.w=mix.w)
  Q <-
    controlvals$optim.penalty * abs(dp$typeI - sig.level)/sqrt(sig.level*(1-sig.level)) +
    controlvals$optim.penalty * abs(dp$power - power)/sqrt(power*(1-power)) +
    dp$EN
  Q
}
```
◇

Fragment referenced in ?.

Uses: `calc.bounds` ?, `oc` ?.

3. Key support functions

"../R/gsDesignOC.R" ?≡

```
#'Operating characteristics of a potential design
#'
#'The values supplied in the function call replace the defaults and a list with all possible #'arguments
#'\code{gsDesignOC} function.
#'
#'@export
#'@param x object of class \code{gsDesignOC}
#'@param EN_theta numeric vector of parameter values for which expected sample size calculation is perfor
#'@param mix.w numeric vector of positive weights for each value of \code{EN_theta}. Defaults to equal we
#'@return a list with the following elements:
#'\describe{
#'\item{typeI}{type I error, ie the probability of not crossing the efficacy boundary under theta=th0}
#'\item{power}{power, ie the probability of crossing the efficacy boundary under theta=thA}
#'\item{EN}{expected sample size under the requested alternatives}
#'}
#'@author Aniko Szabo
#'@keywords design
#'@examples
#'
#'

oc <- function(x, EN_theta=x$thetaA,  mix.w = rep(1, length(EN_theta))){

  if (length(mix.w) != length(EN_theta))
    stop("‘EN_theta‘ and ‘mix.w‘ should have the same length")
  if (!all(mix.w > 0))
    stop("‘mix.w‘ should have only positive elements")

  # under H0
  res0 <- overall.crossprob(x, theta=x$theta0)
  # under Ha
  resA <- overall.crossprob(x, theta=x$thetaA)

  # under EN-alternatives
  en_vec <- numeric(length(EN_theta))
  for (i in seq_along(EN_theta)){
    th <- EN_theta[i]
    # check if we already calculated the EN at this value
    if (abs(th - x$theta0) < .Machine$double.eps){
      en_vec[i] <- res0$EN
    } else if (abs(th - x$thetaA) < .Machine$double.eps) {
      en_vec[i] <- resA$EN
    } else {
      en_vec[i] <- overall.crossprob(x, theta=th)
    }
  }
  en <- en_vec %*% mix.w / sum(mix.w)
  list(typeI = res0$cross.up,
       power = resA$cross.up,
       EN = en)
}
◇
```

File defined by ?, ?, ?, ?, ?, ?.
Defines: oc ?.

"../R/gsDesignOC.R" ?≡

```
#'Control values for gsDesignOC
#'
#'The values supplied in the function call replace the defaults and a list with all possible
#'arguments is returned. The returned list is used as the \code{control} argument to the
#'\code{gsDesignOC} function.
#'
#'@export
#'@param optim.penalty numeric; relative weight of terms ensuring target type I and type II
#'errors versus sample size in optimization routine
#'@return a list with components for each of the possible arguments
#'@author Aniko Szabo
#'@keywords design
#'@examples
#'
#'ocControl(optim.penalty = 100)

ocControl <- function(optim.penalty = 1000){
  list(optim.penalty = optim.penalty)
}
◇
```

File defined by ?, ?, ?, ?, ?, ?.

4. Boundary achieving the desired operating characteristics

The bounds

"../R/gsDesignOC.R" ?≡

```
#'Calculate efficacy/futility boundary values given the timing of analyses and desired operating characte

#'@export
#'@param x an object of class \code{gsDesignOC}

calc.bounds <- function(x){}

◇
```

File defined by ?, ?, ?, ?, ?, ?.
Defines: `calc.bounds` ?.

"../R/gsDesignOC.R" ?≡

```
#'Conversion between nominal significance levels and alternative hypothesis values
#'
#' The \code{convert.bounds} function performs conversion between nominal significance levels
#' for testing eqn{H_0: \theta=\theta_0}{H0: theta=theta0} and the values for the alternatives that have
#' when \eqn{H_0}{H0} is tested at the nominal level.
#'
#' The \code{convert.bounds2} extends this function for both and efficacy and futility boundary,
#' where for the futility boundary the hypothesis eqn{H_A: \theta=\theta_A}{HA: theta=thetaA}  is tested.

#' The power is cumulative over the previous stages.
#
#' Either the first 2 parameters has to be specified, and the other one will be computed.

#'@export
#'@param nominal.level numeric vector (for \code{convert.bounds}) or
```

```
#'  numeric matrix with two rows (for \code{convert.bounds2}) of the nominal significance level for each
#'  When a matrix, row 1 corresponds to the efficacy boundary and row 2 to the futility. Defaults to NULL
#'@param nominal.level numeric vector (for \code{convert.bounds}) or
#'  numeric matrix with two rows (for \code{convert.bounds2}) of the alternative hypotheses for each stag
#'  When a matrix, row 1 corresponds to the efficacy boundary and row 2 to the futility. Defaults to NULL
#'@param n integer vector of sample sizes of each stage. Its sum is the total study sample size.
#'@param power numeric, the target power at each stage.
#'@param theta.null numeric, the null hypothesis being tested
#'@param power.futility numeric, the target power for futility testing at each stage.
#'@param theta.alt numeric, the alternative hypothesis being tested for the futility boundary.

#'@return a list with all the inputs, with the NULL component filled in
#'@author Aniko Szabo
#'@keywords internal
#'@examples
#'
#'(c1 <- convert.bounds(nominal.level = c(0.01, 0.02),
#'                       n = c(30, 50), power=0.8, theta.null=0))
#' convert.bounds(theta = c1$theta, n = c(30, 50), power=0.8, theta.null=0)$nominal.level

convert.bounds <- function(nominal.level=NULL, theta=NULL, n, power, theta.null){
  if (sum(sapply(list(nominal.level, theta), is.null)) !=  1)
    stop("exactly one of 'nominal.level' and 'theta' must be NULL")

  zb <- qnorm(power)


  if (is.null(theta)){
    .pow <- function(th, bounds, ns){
      kk  <- length(bounds)
      gg <- gsDesign::gsProbability(k=kk, theta=th, times =ns, a = rep(-20, kk), b=bounds)
      sum(gg$upper$prob) - power
    }
    k <- length(nominal.level)
    cutoff <- qnorm(nominal.level, lower=FALSE)
    theta <- theta.null + (cutoff + zb)/sqrt(n)  # initial guess
    if (k >=2){
      for (idx in 2:k){
          res <- uniroot(f=.pow, interval=c(0, theta[idx]),
                         bounds=cutoff[1:idx], ns=n[1:idx])
          theta[idx] <- res$root
      }
    }
  } else if (is.null(nominal.level)){
    .pow <- function(x, prev.bounds, ns, theta){
      kk  <- length(prev.bounds) + 1
      gg <- gsDesign::gsProbability(k=kk, theta=theta, n.I =ns,
                          a = rep(-20, kk), b=c(prev.bounds,x))
      sum(gg$upper$prob) - power
    }
    k <- length(theta)
    cutoff <- (theta - theta.null)*sqrt(n) - zb  # initial guess
    if (k >= 2){
      for (idx in 2:k){
        res <- tryCatch(uniroot(f=.pow, interval=c(-20,20),
                       prev.bounds=cutoff[1:(idx-1)], ns=n[1:idx],
                       theta=theta[idx]), error=function(e)e)
        if ("error" %in% class(res)) browser()
        cutoff[idx] <- res$root
      }
    }
    nominal.level <- pnorm(cutoff, lower=FALSE)
  } else if (is.null(n)){
    .pow <- function(x, prev.ns, bounds, theta){
      kk  <- length(prev.ns) + 1
      gg <- gsDesign::gsProbability(k=kk, theta=theta, n.I =c(prev.ns,x),
                          a = rep(-20, kk), b=bounds)
      sum(gg$upper$prob) - power
    }
    k <- length(theta)
    cutoff <- qnorm(nominal.level, lower=FALSE)
    n <- (cutoff + zb)^2 / (theta - theta.null)^2 # initial guess
    if (k >= 2){
      for (idx in 2:k){
        res <- uniroot(f=.pow, interval=c(sum(n[idx-1]), sum(n)*2),
                       prev.ns = n[1:(idx-1)],
                       bounds=cutoff[1:idx],
                       theta=theta[idx])
        n[idx] <- res$root
      }
    }
  }

  res <- list(nominal.level = nominal.level,
              theta = theta,
              n = n,
              cutoff = cutoff,
              theta.null = theta.null,
              power = power)
  res
}

#'@describeIn convert.bounds Conversion between nominal significance and alternatives with both futility
#'@export
#'@keywords internal
#'@examples
#'(c2 <- convert.bounds2(nominal.level = rbind(c(0.01, 0.02), c(0.05,0.1)),
#'                       n = c(30, 50), power=0.8, theta.null=0, power.futility=0.9, theta.alt=0.2))
#' convert.bounds2(theta = c2$theta, n = c(30, 50), power=0.8, theta.null=0,
#'                power.futility=0.9, theta.alt=0.2)$nominal.level

convert.bounds2 <- function(nominal.level=NULL, theta=NULL, n,
                         power, theta.null, power.futility, theta.alt){
  if (sum(sapply(list(nominal.level, theta), is.null)) !=  1)
    stop("exactly one of 'nominal.level' and 'theta' must be NULL")

  zb <- qnorm(power)

  if (is.null(theta)){
    .powU <- function(th, bounds, ns){
      kk  <- NCOL(bounds)
      gg <- gsDesign::gsProbability(k=kk, theta=th, n.I =ns,  a =bounds[2,], b=bounds[1,])
      sum(gg$upper$prob) - power
    }
    .powL <- function(th, bounds, ns){
      kk  <- NCOL(bounds)
      gg <- gsDesign::gsProbability(k=kk, theta=th, n.I =ns,a =bounds[2,], b=bounds[1,])
      sum(gg$lower$prob) - power
    }
    k <- NCOL(nominal.level)
    cutoffs <- qnorm(nominal.level,lower=FALSE)
    thetaU <- theta.null + (cutoffs[1,] + zb)/sqrt(n)  # initial guess
    thetaL <- theta.null + (cutoffs[2,] - zb)/sqrt(n)  # initial guess

    if (k >=2){
      for (idx in 2:k){
        resU <- uniroot(f=.powU, interval=c(-20, 20),
                        bounds=cutoffs[,1:idx,drop=FALSE], ns=n[1:idx])
        thetaU[idx] <- resU$root
        resL <- uniroot(f=.powL, interval=c(-20,20),
                        bounds=cutoffs[,1:idx,drop=FALSE], ns=n[1:idx])
        thetaL[idx] <- resL$root
      }
    }
    theta <- rbind(thetaU, thetaL)
  } else if (is.null(nominal.level)){
    .powU <- function(x, prev.bounds, ns, ths){
      kk  <- NCOL(prev.bounds) + 1
      gg <- gsDesign::gsProbability(k=kk, theta=ths, n.I =ns,
                          a = c(prev.bounds[2,],-20), b=c(prev.bounds[1,],x))
      sum(gg$upper$prob) - power
    }
    .powL <- function(x, prev.bounds, ns, ths){
      kk  <- NCOL(prev.bounds) + 1
      gg <- gsDesign::gsProbability(k=kk, theta=ths, n.I =ns,
                          a = c(prev.bounds[2,],x),
                          b=c(prev.bounds[1,],20))
      sum(gg$lower$prob) - power
    }
    k <- NCOL(theta)
    cutoffs <- matrix(NA, nrow=2, ncol=k)
    cutoffs[1, ] <- (theta[1,] - theta.null)*sqrt(n) - zb  # initial guess
    cutoffs[2, ] <- (theta[2,] - theta.null)*sqrt(n) + zb  # initial guess
    if (k >= 2){
      for (idx in 2:k){
        resU <- tryCatch(uniroot(f=.powU, interval=c(-20,20),
                                 prev.bounds=cutoffs[,1:(idx-1),drop=FALSE], ns=n[1:idx],
                                 ths=theta[1,idx]), error=function(e)e)
        if ("error" %in% class(resU)) browser()
        cutoffs[1,idx] <- resU$root
        resL <- tryCatch(uniroot(f=.powL, interval=c(-20,20),
                                 prev.bounds=cutoffs[,1:(idx-1),drop=FALSE], ns=n[1:idx],
                                 ths=theta[2,idx]), error=function(e)e)
        if ("error" %in% class(resL)) browser()
        cutoffs[2,idx] <- resL$root
      }
    }
    nominal.level <- pnorm(cutoffs, lower.tail=FALSE)
  }

  res <- list(nominal.level = nominal.level,
              theta = theta,
              n = n,
              cutoffs = cutoffs,
              theta.null = theta.null,
              theta.alt = theta.alt,
              power = power)
  res
}
◇
```

File defined by ?, ?, ?, ?, ?, ?.

"../R/gsDesignOC.R" ?≡

```
#'Overall probability of crossing the efficacy and futility boundaries
#'
#'The \code{overall.crossprob} calculates the probability of crossing the efficacy and, optionally,
#' the futility bound, given a series of alternative (and optionally, null) hypotheses tested at
#' each stage.

#'@export
#'@inheritParams gsDesignOC
#'@param theta numeric; value of the parameter under which the crossing probabilities are computed
#'@param timing numeric; cumulative proportio
#'@param
#'@param
#'@param
#'@param
#'@param
#'@param
#'@return a list with all the inputs, with the NULL component filled in
#'@author Aniko Szabo
#'@keywords internal
#'@examples
#'
#'(c1 <- convert.bounds(nominal.level = c(0.01, 0.02),
#'                        n = c(30, 50), power=0.8, theta.null=0))
#' convert.bounds(theta = c1$theta, n = c(30, 50), power=0.8, theta.null=0)$nominal.level
overall.crossprob <- function(theta, timing, n, thA.seq, power, th0.seq=NULL, power.futility=NULL,
                              sig.level_final=NULL){
  k <- length(thA.seq)
  bb <- convert2.cum(thetas=rbind(thA.seq, th0.seq), n=n*timing, power=power)$cutoffs
  if (!is.null(sig.level_final)){
    bb.last <- qnorm(sig.level_final, lower.tail = FALSE)
    bb <- cbind(bb, rep(bb.last,2))
    timing <- c(timing, 1)
    k <- k + 1
  }
  p <- gsProbability(k=k, theta=theta, n.I=timing*n,
                     a=bb[2,], b=bb[1,])
  list(cross.up=colSums(p$upper$prob), cross.down=colSums(p$lower$prob),
       en=p$en, bounds=bb)
}
overall.crossprob.alt <- function(th.seq, timing, n, theta=0,
                                   sig.level_final=.alpha*0.8, power=1-.beta){
  k <- length(th.seq)
  bb <- convert.alt.cum(theta=th.seq, n=n*timing, power=power)$cutoff
  if (!is.null(sig.level_final)){
    bb.last <- qnorm(sig.level_final, lower.tail = FALSE)
    bb <- c(bb, bb.last)
    timing <- c(timing, 1)
    k <- k + 1
  }
  p <- gsProbability(k=k, theta=theta, n.I=timing*n,
                     a=rep(-20,k), b=bb)
  list(cross.p=colSums(p$upper$prob), en=p$en, upper=bb)
}
```

◇

File defined by ?, ?, ?, ?, ?, ?.

5. Utility help-functions

"../R/Utility.R" ?≡

```
#'One-sample Z-based sample size for one-sided test
#'
#'The \code{ztest.n} function finds the sample size of a one-sided Z-test with given power
#'and significance level
#'
#'@param delta numeric; targeted effect size
#'@param sd numeric; standard deviation, assumed known
#'@param sig.level numeric; one-sided significance level
#'@param power numeric; target power
#'@return numeric value with the (fractional) sample size achieving the target power
#'@author Aniko Szabo
#'@keywords internal
#'@examples
#'
#'ztest.n(delta=1, sd=1, sig.level=0.05, power=0.9)

ztest.n <- function(delta, sd, sig.level, power){
  za <- qnorm(sig.level, lower=FALSE)
  zb <- qnorm(power)
  n <- (za+zb)^2 * sd^2/delta^2
  n
  }
◇
```

File defined by ?, ?.
Defines: `ztest.n` ?.

"../R/Utility.R" ?≡

```
#'Conversion between boundary and nominal significance level
#'
#'The \code{nom.to.bnd} and \code{bnd.to.nom} functions perform conversion between the boundary and
#' matching significance level.
#'
#'@param nominal.level numeric vector or matrix with two rows of the nominal significance level for each
#'  When a matrix, row 1 corresponds to the efficacy boundary and row 2 to the futility. Defaults to NULL
#'@param cutoffs numeric vector or numeric matrix with two rows of the z-test cutoffs for each stage.
#'  When a matrix, row 1 corresponds to the efficacy boundary and row 2 to the futility. Defaults to NULL
#'@param n integer vector of sample sizes of each stage. Its sum is the total study sample size.
#'@param theta.null numeric, the null hypothesis being tested
#'@param theta.alt numeric, the alternative hypothesis being tested for the futility bound calculation
#'@keywords internal
#'@examples
#'
#'## only efficacy
#'(b1 <- nom.to.bnd(nominal.level = c(0.01, 0.02), n = c(30, 50)))
#'bnd.to.nom(b1, n=c(30,50))

nom.to.bnd <- function(nominal.level, n, theta.null, theta.alt=NULL){
  nominal.level <- rbind(nominal.level) # raises to matrix if one row
  m <- nrow(nominal.level)
  k <- ncol(nominal.level)
  cutoffs <- matrix(NA, nrow=m, ncol=k)
  cutoffs[1, ] <- qnorm(nominal.level[1,], lower=FALSE)
  if (k > 1) {
    # flip direction + shift hypothesis
```

```
      cutoffs[2, ] <- qnorm(nominal.level[2,], lower=TRUE) + sqrt(n)*(theta.alt -theta.null)
      }
    cutoffs[1:k,]  # drops to vector if one row
  }

  #'@describeIn nom.to.bnd Conversion from nominal significance level to boundary
  #'@export
  #'@keywords internal
  #'@examples
  #'## both efficacy and futility
  #'(b2 <- nom.to.bnd(nominal.level = rbind(c(0.01, 0.02), c(0.05,0.1)),
  #'                  n = c(30, 50)))
  #'bnd.to.nom(b2, n=c(30,50))

  bnd.to.nom <- function(cutoffs, n, theta.null, theta.alt=NULL){
    cutoffs <- rbind(cutoffs) # raises to matrix if one row
    m <- nrow(cutoffs)
    k <- ncol(cutoffs)
    noms <- matrix(NA, nrow=m, ncol=k)
    noms[1,] <- pnorm(cutoffs[1,], lower=FALSE)
    if (k > 1) {
      # flip direction + shift hypothesis
      noms[2,] <- pnorm(cutoffs[2,] - sqrt(n)*(theta.alt-theta.null), lower=TRUE)
      }
    noms[1:k,] # drops to vector if one row
  }
```

◇
File defined by ?, ?.