Design and analyze a single-arm group sequential test with binary endpoints using 'BinGSD'

Lei Wang

October 11, 2019

Let $X_1, X_2 \ldots$ be binary outcomes following Bernoulli distribution $b(1,p)$, where 1 stands for the case that the patient responses to the treatment and 0 otherwise. Consider an at-most-K-stage group sequential design with the null hypothesis $H_0: p = p_0$ and n_k samples available at analysis k, $k = 1,2, \ldots, K$. For $k \leq K-1$, the trial stops if the test statistic Z_k crosses the lower boundary (or the futility boundary), that is, $Z_k \leq l_k$. Otherwise, more samples will be collected, and the trial would go to the $(k+1)^{th}$ analysis or finally stops at the K^{th} analysis. To ensure termination at the end of trial, the lower boundary for the final analysis l_K is set to equal the corresponding upper boundary u_K. At the final stage, we reject H_0 if the test statistic $Z_K \geq u_K$, where the upper boundary (or the efficacy boundary) u_K is determined by the desired overall type I error rate.

Let I_k be the Fisher information at analysis k, $k = 1,2, \ldots, K$. For binomial distribution, $I_k = \frac{n_k}{p(1-p)}$ which is proportional to the sample size available at that time point. Then the maximum Fisher information would be achieved at the final analysis, that is, $I_{max} = I_K$. Denote the information fraction at analysis k as $t_k = \frac{I_k}{I_{max}} = \frac{n_k}{n_K}$.

The rest of the document is organized as follows. Section 1 illustrates the procedure of designing the group-sequential trial under asymptotic test and exact test. More specifically, subsection 1.1 presents two algorithms for the two tests, respectively, and the rest two subsections show how to calculate the probabilities in Algorithm 1 and 2. Section 2 covers the methodology for calculating conditional power given some interim results. Section 3 discusses some issues encountered during the draft of this document. The last section provides step-by-step guidance to use the package.

Section 1 Calculate sample size and boundaries for the group-sequential test

Section 1.1 Algorithm 1 and 2

During the trial, the only one efficacy bound is set for the last analysis. Thus, type I error is spent its entirety at the last look or equally the interim upper boundaries are set to be ∞. The procedure to determine sample size and boundaries is analogous to that in `East 6`[1]. To make the boundaries non-binding, the lower boundaries are sequentially computed given the upper boundary. If the requirement for power is not satisfied with the current design, the maximum sample size n_K will be increased but all the boundaries remain unchanged. Define a monotone beta spending function $\beta(t)$ with $\beta(0) = 0\ and\ \beta(1) = \beta$. The algorithms searching for sample size and boundary at each analysis work as follows:

Algorithm 1. The searching procedure for asymptotic test.

Input: the maximum number of analyses K; the information fractions t_k; the desired overall type II error level β with the proportions spent at each analysis; the desired overall type I error level α; the null hypothesis p_0; specific alternative p_1.

Output: the sample sizes n_k; lower boundaries $\{l_1, l_2, \dots, l_{K-1}\}$; the upper boundary u_K; the actual overall type I error; the actual type II error at each analysis; power of the group sequential test.

Initialization. Calculate the upper boundary u_K such that $P_{p_0}\{Z_K \geq u_K\} = \alpha$. Find the value n_K that satisfies the type II error equation $P_{p_1}\{Z_K < u_K\} = \beta$.

Step 1. Calculate the sample sizes n_k according to n_K and the information fractions.

Step 2. At the first look, calculate the lower boundary l_1 such that $P_{p_1}\{Z_1 \leq l_1\} = \beta(t_1)$.

Step 3. For the subsequent looks $k = 2,3, \dots, K-1$, having already computed the lower boundaries $\{l_1, l_2, \dots, l_{k-1}\}$, find the lower boundary l_k such that $P_{p_1}\{Z_1 \leq l_1\} + P_{p_1}\{Z_1 > l_1, Z_2 \leq l_2\} + \cdots + P_{p_1}\{Z_1 > l_1, \cdots, Z_{k-1} > l_{k-1}, Z_k \leq l_k\} = \beta(t_k)$.

Step 4. Set $l_K = u_K$ to ensure that a decision can be made at the last analysis. Then the power of the design is

$$1-\beta^* = P_{p_1}\{Z_1 > l_1, \cdots, Z_{K-1} > l_{K-1}, Z_K \geq u_K\}$$
$$= 1 - P_{p_1}\{Z_1 \leq l_1\} - P_{p_1}\{Z_1 > l_1, Z_2 \leq l_2\} - \cdots$$
$$- P_{p_1}\{Z_1 > l_1, \cdots, Z_{K-1} > l_{K-1}, Z_K \leq u_K\}.$$

Step 5. If $1-\beta^* \geq 1-\beta$, stop the algorithm and output the derived boundaries as well as the sample sizes. Otherwise, set $n_K = n_K + 1$ and go to Step 4.

Note that under null hypothesis, Z_K follows standard normal distribution. Thus, in **Initialization**, u_K is $1-\alpha$ quantile of $N(0,1)$, which is not relevant to the maximum sample size n_K. Given u_K, n_K is determined under the alternative hypothesis and the resulting n_K is probably not an integer. While coding, we round all the sample sizes up to whole numbers before calculating boundary crossing probabilities.

For looks $k = 2,3, \ldots, K-1$, we use root finding algorithm to sequentially search for the lower bounds. Since u_K is fixed and lower bound for the next analysis cannot be less than the former, it is natural to set initial search interval within $[l_{k-1}, u_K]$. Bisection method narrows the search interval quickly, but if may converge slowly if the solution falls very close to either of the two endpoints of the search interval. False position method speeds up the algorithm by using the weighted average of two endpoints as the endpoint in the next search. However, with wide search interval, false position method may fail to achieve convergence. In our package, we first use bisection method twice to narrow the search interval. If the actual type II error is unsatisfactory, implement false position method until convergence is achieved or the iteration times reached 30.

Algorithm 2. The searching procedure for exact test.

Input: the maximum number of analyses K; the information fractions t_k; the desired overall type II error level β with the proportions spent at each analysis; the desired overall type I error level α; the null hypothesis p_0; specific alternative p_1; n_K derived from asymptotic test.

Output: the sample sizes n_k; lower boundaries $\{l_1, l_2, \ldots, l_{K-1}\}$; the upper boundary u_K; the actual type I error α^*; the actual type II error spent at each analysis; power of the group sequential test.

Initialization. Use n_K derived from Algorithm 1 as the starting value.

Step 0. Given n_K, find the smallest integer u_K such that $\alpha^* = P_{p_0}\{Z_K \geq u_K\} \leq \alpha$. If $P_{p_1}\{Z_K < u_K\} \leq \beta$ holds, go to Step 1. Otherwise, progressively increase n_K with increment 1 until the constraints on two types of errors are both satisfied.

Step 1. Calculate the sample sizes n_k according to n_K and the information fractions.

Step 2. At the first look, find the largest integer l_1 such that $P_{p_1}\{Z_1 \leq l_1\} \leq \beta(t_1)$.

Step 3. For the subsequent looks $k = 2,3, \ldots, K-1$, having already computed the lower boundaries $\{l_1, l_2, \ldots, l_{k-1}\}$, find the largest integer l_k such that $P_{p_1}\{Z_1 \leq l_1\} + P_{p_1}\{Z_1 > l_1, Z_2 \leq l_2\} + \cdots + P_{p_1}\{Z_1 > l_1, \cdots, Z_{k-1} > l_{k-1}, Z_k \leq l_k\} \leq \beta(t_k)$.

Step 4. Set $l_K = u_K$ to ensure that a decision can be made at the last analysis. Then the power of the design is

$$\begin{aligned} 1 - \beta^* &= P_{p_1}\{Z_1 > l_1, \cdots, Z_{K-1} > l_{K-1}, Z_K \geq u_K\} \\ &= 1 - P_{p_1}\{Z_1 \leq l_1\} - P_{p_1}\{Z_1 > l_1, Z_2 \leq l_2\} - \cdots \\ &\quad - P_{p_1}\{Z_1 > l_1, \cdots, Z_{K-1} > l_{K-1}, Z_K < u_K\}. \end{aligned}$$

Step 5. If $1 - \beta^* \geq 1 - \beta$, stop the algorithm and output the derived boundaries as well as the sample sizes.

Otherwise, set $n_K = n_K + 1$ and go to Step 0.

Initialization follows the strategy in Section 12.1.2 of [4]. The calculation of boundary crossing probabilities here borrowed strength from the source code of function `gsBinomialExact` in package 'gsDesign'[3].
R basic function `qbinom(p, size, prob, lower.tail = TRUE, log.p = FALSE)` returns the quantile which is defined as the smallest value y such that P(Y $\leqslant$ y)$\geqslant$ p. In Step 0, u_K is the smallest integer such that $\alpha^* = P_{p_0}\{Z_K \geq u_K\} = 1 - P_{p_0}\{Z_K < u_K\} \leq \alpha$. From another point of view, u_K should be the smallest integer such that $P_{p_0}\{Z_K < u_K\} = P_{p_0}\{Z_K \leq u_K - 1\} \geq 1 - \alpha$. Thus, $u_K = qbinom(1 - \alpha, n_K, p_0) + 1$. Similarly, $P_{p_1}\{Z_K < u_K\} = P_{p_1}\{Z_K \leq u_K - 1\} = pbinom(u_K - 1, n_K, p_1)$.

While searching for the first lower bound in **Step 2**, we are trying to find the largest integer l_1 such that $P_{p_1}\{Z_1 \leq l_1\} \leq \beta(t_1)$. $qbinom(\beta(t_1), n_1, p_1)$ returns

the smallest integer $temp1$ that makes $P_{p_1}\{Z_1 \leq temp1\} \geq \beta(t_1)$ hold. Thus, integer $temp2 = temp1 - 1$ has the property that $P_{p_1}\{Z_1 \leq temp2\} < \beta(t_1)$. As a result, l_1 has a value identical to one of $temp2$ and $temp1$.

Note that in **Step 3**, the type II error that does not spent by the first k analyses will be carried over for the use of the following analysis. The reason why we carry over unspent type II error can be found in Section 3.

For Algorithm 1, the boundaries can be any real value within the searching interval. Therefore, we combine bisection with false position method to accelerate the convergence. However, for exact test, l_k can only be an integer within l_{k-1} and u_K. With definite number of possible solutions, bisection can narrow the search interval to an interval containing only two integers after a few iterations. For example, if the starting interval contains 16000 integers, the length of searching interval reaches 2 after only 13 bisections. Therefore, we adopt the bisection when finding the lower bounds in Algorithm 2.

Due to the discreteness of binomial distribution, unlike asymptotic test, the sequence of lower bounds is strictly increasing. To make sure $l_k < u_K$, in hidden function bound2 of the package, a constraint is set that $l_k = u_K$ occurs only when $k = K - 1$, otherwise, the program throws out an error message. Note that if $l_{K-1} = u_K$ holds, the last upper bound will be crossed with probability equal to 1 after passing the analysis $K - 1$.

In function bound2, the iteration stops when the length of searching interval reaches 2, that is, the lower search bound is 1 less than the upper searching bound. During the iteration, the new searching bound is updated with the middle value between lower and upper searching bound via R basic function `floor`. If the difference between two search bounds is odd like 3, 5, 7, …, then the middle value is not an integer. In this case, `floor`(the middle value)>lower searching bound, so it is a valid value for a new search bound. When the difference is even, like 2, 4, 6, …, the middle value is already an integer and `floor`(the middle value) will not change anything.

Section 1.2 The probabilities in Algorithm 2 based on binomial distribution

With exact test, the test statistic at analysis k can be $Z_k = \sum_{s=1}^{n_k} X_s$ which follows binomial distribution $B(n_k, p)$. It is easy to see

$$P_{p_1}\{Z_1 \le l_1\} = \sum_{z_1=0}^{l_1} C_{n_1}^{z_1}(p_1)^{z_1}(1-p_1)^{n_1-z_1}, \quad (1)$$

For $k = 2,3,\dots,K$, the joint distribution of $Z_1, Z_2, \dots, Z_k$ under the alternative hypothesis is

$$P_{p_1}\{Z_1 = z_1, \cdots, Z_{k-1} = z_{k-1}, Z_k = z_k\} = P_{p_1}\left\{\sum_{s=1}^{n_1} X_s = z_1, \cdots, \sum_{s=1}^{n_{k-1}} X_s = z_{k-1}, \sum_{s=1}^{n_k} X_s = z_k\right\} = P_{p_1}\left\{\sum_{s=1}^{n_1} X_s = z_1, \sum_{s=n_1+1}^{n_2} X_s = z_2 - z_1, \cdots, \sum_{s=n_{k-1}+1}^{n_k} X_s = z_k - z_{k-1}\right\} = P_{p_1}\left\{\sum_{s=1}^{n_1} X_s = z_1\right\} \prod_{m=2}^{k} P_{p_1}\{\sum_{s=n_{m-1}+1}^{n_m} X_s = z_m - z_{m-1}\} = C_{n_1}^{z_1}(p_1)^{z_1}(1-p_1)^{n_1-z_1} \prod_{m=2}^{k}[C_{n_m-n_{m-1}}^{z_m-z_{m-1}}(p_1)^{z_m-z_{m-1}}(1-p_1)^{n_m-n_{m-1}-z_m+z_{m-1}}], \quad (2)$$

Recall that $z_k \ge l_{k-1} + 1$ always holds otherwise the trial would stop before analysis k. Therefore, the probability

$$P_{p_1}\{Z_1 > l_1, \cdots, Z_{k-1} > l_{k-1}, Z_k \le l_k\} = \sum_{z_k=l_{k-1}+1}^{l_k} \sum_{z_{k-1}=\max\{l_{k-1}+1, z_k-n_k+n_{k-1}\}}^{\min\{n_{k-1}, z_k\}} \cdots \sum_{z_1=\max\{l_1+1, z_2-n_2+n_1\}}^{\min\{n_1, z_2\}} C_{n_1}^{z_1}(p_1)^{z_1}(1-p_1)^{n_1-z_1} \prod_{m=2}^{k}[C_{n_m-n_{m-1}}^{z_m-z_{m-1}}(p_1)^{z_m-z_{m-1}}(1-p_1)^{n_m-n_{m-1}-z_m+z_{m-1}}], \quad (3)$$

Define $l_k^*(z_k) = \max\{l_{k-1}+1, z_k - n_k + n_{k-1}\}$ and $u_k^*(z_k) = \min\{n_{k-1}, z_k\}$. Also, let $C_1(z_1, p) = C_{n_1}^{z_1}(p)^{z_1}(1-p)^{n_1-z_1}$ and $C_k(z_k, p) = \sum_{z_{k-1}=l_k^*(z_k)}^{u_k^*(z_k)} C_{k-1}(z_{k-1}, p)\,[C_{n_k-n_{k-1}}^{z_k-z_{k-1}}(p)^{z_k-z_{k-1}}(1-p)^{n_k-n_{k-1}-z_k+z_{k-1}}]$. As in Section 12.1.2 of [4], the probability of crossing the lower boundary at stage k is

$$P_{p_1}\{Z_1 > l_1, \cdots, Z_{k-1} > l_{k-1}, Z_k \le l_k\} = \sum_{z_k=l_{k-1}+1}^{l_k} C_k(z_k, p_1) = \sum_{z_k=l_{k-1}+1}^{l_k} \sum_{z_{k-1}=l_k^*(z_k)}^{u_k^*(z_k)} C_{k-1}(z_{k-1}, p_1)\,[C_{n_k-n_{k-1}}^{z_k-z_{k-1}}(p_1)^{z_k-z_{k-1}}(1-p_1)^{n_k-n_{k-1}-z_k+z_{k-1}}], \quad (4)$$

In **Step 0** of Algorithm 2, the probability

$$P_{p_0}\{Z_K \ge u_K\} = \sum_{z_K=u_K}^{n_K} C_{n_K}^{z_K}(p_0)^{z_K}(1-p_0)^{n_K-z_K}, \quad (5)$$

The actual overall type II error is

$$\beta^* = P_{p_1}\{Z_1 \le l_1\} + P_{p_1}\{Z_1 > l_1, Z_2 \le l_2\} + \cdots + P_{p_1}\{Z_1 > l_1, \cdots, Z_{K-1} > l_{K-1}, Z_K \le u_K\} = \sum_{z_1=0}^{l_1} C_1(z_1, p_1) + \sum_{k=2}^{K} \sum_{z_k=l_{k-1}+1}^{l_k} C_k(z_k, p_1), \quad (6)$$

Section 1.3 The probabilities in Algorithm 1 based on normal distribution

For a p_0 not close to 1 or 0, with a large sample size, the test statistic at analysis k can be $Z_k = \hat{\theta}_k\sqrt{I_k}$ which asymptoticly follows normal distribution $N(\theta\sqrt{I_k}, 1)$, where $\theta = p - p_0$ and $\hat{\theta}_k = \hat{p}_k - p_0 = \sum_{s=1}^{n_k} \frac{X_s}{n_k} - p_0$. It is worth mentioning that I_k is the inverse of the variance $var(\hat{\theta}_k)$.

Under H_0, $\theta = p_0 - p_0 = 0$, and

$$P_{p_0}\{Z_K \geq u_K\} = 1 - \Phi(u_K), \tag{7}$$

where $\Phi(\cdot)$ is the cumulative distribution function of standard normal distribution. In **Initialization** of Algorithm 1, $P_{p_0}\{Z_K \geq u_K\} = \alpha$. Thus, $u_K = \Phi^{-1}(1-\alpha)$. To obtain the initial maximum sample size n_K, let

$$\begin{aligned}\beta = P_{p_1}\{Z_K < u_K\} &= P_{p_1}\left\{Z_K - \theta\sqrt{I_K} < u_K - \theta\sqrt{I_K}\right\} \\ &= \Phi\left(u_K - (p_1 - p_0)\sqrt{\frac{n_K}{p_1(1-p_1)}}\right)\end{aligned}$$

Thus, $n_K = p_1(1-p_1)[(u_K - \Phi^{-1}(\beta))/(p_1 - p_0)]^2$. Let the initial maximum sample size be $n_K = [n_K]$, the sequence of sample sizes at each analysis be $\{[n_K t_1], \dots, [n_{K-1}t_{K-1}], [n_K]\}$, where [] rounds the value up to its nearest whole value.

In **Step 1** of Algorithm 1,

$$P_{p_1}\{Z_1 \leq l_1\} = P_{p_1}\left\{Z_1 - \theta\sqrt{I_1} \leq l_1 - \theta\sqrt{I_1}\right\} = \Phi\left(l_1 - (p_1 - p_0)\sqrt{\frac{n_1}{p_1(1-p_1)}}\right), \tag{8}$$

Then $l_1 = \Phi^{-1}\big(\beta(t_1)\big) + (p_1 - p_0)\sqrt{\frac{n_1}{p_1(1-p_1)}}$.

For the sequence of test statistics $\{Z_1, Z_2, \dots, Z_k\}$, $k = 1,2, \dots, K$, it is easy to check that:

(1) $(Z_1, Z_2, \dots, Z_k)$ is multivariate normal;
(2) $E(Z_k) = \theta\sqrt{I_k}$;
(3) $COV\big(Z_{k_1}, Z_{k_2}\big) = \sqrt{I_{k_1}/I_{k_2}}, 1 \leq k_1 \leq k_2 \leq K$.

The joint normal distribution of $\{Z_1, Z_2, \dots, Z_K\}$ has a mean vector $\boldsymbol{\mu} = \{\theta\sqrt{I_1}, \theta\sqrt{I_2}, \dots, \theta\sqrt{I_K}\}$ and a covariance matrix

$$\Sigma_{K\times K} = \begin{bmatrix} 1 & \sqrt{I_1/I_2} & \sqrt{I_1/I_3} & \cdots & \sqrt{I_1/I_K} \\ \sqrt{I_1/I_2} & 1 & \sqrt{I_2/I_3} & \cdots & \sqrt{I_2/I_K} \\ \sqrt{I_1/I_3} & \sqrt{I_2/I_3} & 1 & \cdots & \vdots \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ \sqrt{I_1/I_K} & \sqrt{I_2/I_K} & \sqrt{I_3/I_K} & \cdots & 1 \end{bmatrix}$$

The probability

$$P_{p_1}\{Z_1 > l_1, \cdots, Z_{k-1} > l_{k-1}, Z_k \le l_k\} =$$
$$\int_{l_1}^{+\infty} \cdots \int_{l_{k-1}}^{+\infty} \int_{-\infty}^{l_k} f_{Z_1,Z_2,\dots,Z_k}(z_1, z_2, \dots, z_k; \theta) d_{z_k} d_{z_{k-1}} \cdots d_{z_1} =$$
$$\int_{l_1}^{+\infty} \cdots \int_{l_{k-1}}^{+\infty} \int_{-\infty}^{l_k} \int_{-\infty}^{+\infty} \cdots \int_{-\infty}^{+\infty} f_{Z_1,Z_2,\dots,Z_K}(z_1, z_2, \dots, z_K; \theta) d_{z_K} d_{z_{K-1}} \cdots d_{z_1}, \qquad (9)$$

where $f_{Z_1,Z_2,\dots,Z_k}(z_1, z_2, \dots, z_k; \theta)$ and $f_{Z_1,Z_2,\dots,Z_K}(z_1, z_2, \dots, z_K; \theta)$ are joint density function of $\{Z_1, Z_2, \dots, Z_k\}$ and $\{Z_1, Z_2, \dots, Z_K\}$, respectively. Given the lower bounds $\{l_1, l_2, \dots, l_{k-1}\}$, let $P_{p_1}\{Z_1 > l_1, \cdots, Z_{k-1} > l_{k-1}, Z_k \le l_k\} = \beta(t_k) - \beta(t_{k-1})$, l_k can be determined by search method.

With $l_K = u_K$, the actual overall type II error is

$$\beta^* = P_{p_1}\{Z_1 \le l_1\} + P_{p_1}\{Z_1 > l_1, Z_2 \le l_2\} + \cdots + P_{p_1}\{Z_1 > l_1, \cdots, Z_{K-1} > l_{K-1}, Z_K \le u_K\}, \qquad (10)$$

Further, the actual type I error is

$$\alpha^* = 1 - P_{p_0}\{Z_1 \le l_1\} - P_{p_0}\{Z_1 > l_1, Z_2 \le l_2\} - \cdots - P_{p_0}\{Z_1 > l_1, \cdots, Z_{K-1} > l_{K-1}, Z_K \le u_K\}. \qquad (11)$$

With formula (11), the computation of α^* can be easily done by a procedure similar to that for (10).

Section 2 Calculate conditional power

The conditional power quantifies the conditional probability of rejecting the null hypothesis at the final analysis, given the current interim result. Assume $1 \le i < K$, having obtained the boundaries and the sample sizes, consider the conditional

probability of crossing the upper boundary at analysis K given information available up to and including analysis i.

$$\alpha_{i,K}(p|z_i) = P_p\{Z_K \geq u_K, \cap_{m=i+1}^{K-1}\{l_m < Z_m\}|Z_i = z_i\}, \quad (12)$$

For single-arm group sequential design with binary endpoint, two types of test are available-exact test and asymptotic test.

(1) Exact test based on binomial distribution. The test statistic at analysis k can be $Z_k = \sum_{s=1}^{n_k} X_s$ which follows binomial distribution $B(n_k, p)$.

For $1 \leq i < j \leq K$, define increment $Z_{i,j} = Z_j - Z_i = \sum_{s=n_i+1}^{n_j} X_s$. The increment statistic $Z_{i,j}$ follows the binomial distribution $B(n_j - n_i, p)$ independently of Z_i. Then $Z_j = Z_{i,j} + Z_i$. Given interim results at analysis i, the conditional probability of crossing the upper boundary at analysis K is

$$\begin{aligned}\alpha_{i,K}(p|z_i) &= P_p\{Z_K \geq u_K, \cap_{m=i+1}^{K-1}\{l_m < Z_m\}|Z_i = z_i\} \\ &= P_p\{Z_{i,K} + Z_i \geq u_K, \cap_{m=i+1}^{K-1}\{l_m < Z_{i,m} + Z_i\}|Z_i = z_i\} \\ &= P_p\left\{Z_{i,K} \geq u_K - z_i, \cap_{m=i+1}^{K-1}\{l_m - z_i < Z_{i,m}\}\right\}, \quad (13)\end{aligned}$$

Let $u_K^*(z_i) = u_K - z_i$, $l_m^*(z_i) = l_m - z_i$. Note that the definitions of $l_m^*(z_i)$ and $u_K^*(z_i)$ here are different from those in Section 1.2. Formula (13) can be restated as

$$\alpha_{i,K}(p|z_i) = P_p\left\{Z_{i,K} \geq u_K^*(z_i), \cap_{m=i+1}^{K-1}\{l_m^*(z_i) < Z_{i,m}\}\right\} \quad (14)$$

The joint density function of $Z_{i,m}, m = i+1, i+2, \ldots, K$ is:

$$P_p\left\{\sum_{s=n_i+1}^{n_K} X_s = z_{i,K}, \sum_{s=n_i+1}^{n_{K-1}} X_s = z_{i,K-1}, \dots, \sum_{s=n_i+1}^{n_{i+2}} X_s = z_{i,i+2}, \sum_{s=n_i+1}^{n_{i+1}} X_s = z_{i,i+1}\right\}$$

$$= P_p\left\{\sum_{s=n_{K-1}+1}^{n_K} X_s = z_{i,K} - z_{i,K-1}, \sum_{s=n_{K-2}+1}^{n_{K-1}} X_s = z_{i,K-1} - z_{i,K-2}, \dots, \sum_{s=n_{i+1}+1}^{n_{i+2}} X_s = z_{i,i+2} - z_{i,i+1}, \sum_{s=n_i+1}^{n_{i+1}} X_s = z_{i,i+1}\right\}$$

$$= P_p\left\{\sum_{s=n_i+1}^{n_{i+1}} X_s = z_{i,i+1}\right\} \prod_{m=i+2}^{K} P_p\left\{\sum_{s=n_{m-1}+1}^{n_m} X_s = z_{i,m} - z_{i,m-1}\right\} \quad (15)$$

Define $l_m^{**}(z_{i,m}) = \max\{l_{m-1}^*(z_i) + 1, z_{i,m} - n_m + n_{m-1}\}$ and $u_m^{**}(z_{i,m}) = \min\{n_{m-1} - n_i, z_{i,m}\}, m = i+2, i+3, \dots, K$. Besides, let $C_{i+1}(z_{i,i+1}, p) = C_{n_{i+1}-n_i}^{z_{i,i+1}}(p)^{z_{i,i+1}}(1-p)^{n_{i+1}-n_i-z_{i,i+1}}$ and $C_m(z_{i,m}, p) = \sum_{z_{i,m-1}=l_m^{**}(z_{i,m})}^{u_m^{**}(z_{i,m})} C_{m-1}(z_{i,m-1}, p)\,[C_{n_m-n_{m-1}}^{z_{i,m}-z_{i,m-1}}(p)^{z_{i,m}-z_{i,m-1}}(1-p)^{n_m-n_{m-1}-z_{i,m}+z_{i,m-1}}]$. Combine (14) and (15), we have,

$$\alpha_{i,K}(p|z_i) = P_p\{\textstyle\sum_{s=n_i+1}^{n_K} X_s \geq u_K^*(z_i), l_{K-1}^*(z_i) < \sum_{s=n_i+1}^{n_{K-1}} X_s, \dots, l_{i+2}^*(z_i) < \sum_{s=n_i+1}^{n_{i+2}} X_s, l_{i+1}^*(z_i) < \sum_{s=n_i+1}^{n_{i+1}} X_s\} = \sum_{z_{i,K}=u_K^*(z_i)}^{n_K-n_i} C_K(z_{i,K}, p), \quad (16)$$

Given (16), $\alpha_{i,K}(p|z_i)$ can be calculated using the same process as the unconditional type I error of the design.

(2) Asymptotic test based on normal distribution. The test statistic at analysis k can be $Z_k = \hat{\theta}_k\sqrt{I_k}$ which asymptoticly follows normal distribution $N(\theta\sqrt{I_k}, 1)$, with $\theta = p - p_0$ and $\hat{\theta}_k = \hat{p}_k - p_0 = \sum_{s=1}^{n_k} \frac{X_s}{n_k} - p_0$. Following

P8892 of [2], in practice, I_k in Z_k can be derived by the estimated sample proportion $\hat{p}_k$.

Given interim results at analysis i, the conditional probability of crossing the upper boundary at the final analysis is

$$\alpha_{i,K}(p|z_i) = P_p\{Z_K \geq u_K, \cap_{m=i+1}^{K-1}\{l_m < Z_m\}|Z_i = z_i\}$$
$$= P_p\{Z_K \geq u_K, l_{K-1} < Z_{K-1}, \ldots, l_{i+1} < Z_{i+1}|Z_i = z_i\}, \quad (17)$$

Recall Section 1.3, with prespecified information sequence $\{I_1, I_2, \ldots, I_K\}$, $Z_k\sqrt{I_k} - Z_{k-1}\sqrt{I_{k-1}}$ follows the normal distribution $N(\theta\Delta_k, \Delta_k)$ independently of $\{Z_1, Z_2, \ldots, Z_{k-1}\}$, where $\Delta_k = I_k - I_{k-1}$ for $k = 2, \ldots, K$. Analogously, for $j > k$, $Z_{k,j} = Z_j\sqrt{I_j} - Z_k\sqrt{I_k} = \left(\sum_{s=1}^{n_j}\frac{X_s}{n_j} - p_0\right)\frac{n_j}{p(1-p)} - \left(\sum_{s=1}^{n_k}\frac{X_s}{n_k} - p_0\right)\frac{n_k}{p(1-p)}$

$$= \frac{\sum_{s=1}^{n_j}X_s - n_j p_0}{p(1-p)} - \frac{\sum_{s=1}^{n_k}X_s - n_k p_0}{p(1-p)} = \frac{\sum_{s=n_k+1}^{n_j}X_s - (n_j - n_k)p_0}{p(1-p)}$$
$$= \left(\frac{\sum_{s=n_k+1}^{n_j}X_s}{(n_j - n_k)} - p_0\right)\frac{(n_j - n_k)}{p(1-p)} = \left(\frac{\sum_{s=n_k+1}^{n_j}X_s}{(n_j - n_k)} - p_0\right)(I_j - I_k)$$

$Z_{k,j}$ follows the normal distribution $N\big(\theta(I_j - I_k), (I_j - I_k)\big)$ independently of $\{Z_1, Z_2, \ldots, Z_{k-1}\}$. Z_j can be rewritten as $Z_j = \frac{Z_{k,j} + Z_k\sqrt{I_k}}{\sqrt{I_j}}$. For $j > k > i$,

$$COV\big(Z_{i,j}, Z_{i,k}\big) = \frac{(I_j - I_i)(I_k - I_i)}{(n_j - n_i)(n_k - n_i)}COV\left(\sum_{s=n_i+1}^{n_j}X_s, \sum_{s=n_i+1}^{n_k}X_s\right)$$
$$= \frac{1}{p^2(1-p)^2}D\left(\sum_{s=n_i+1}^{n_k}X_s\right) = \frac{(n_k - n_i)}{p(1-p)} = (I_k - I_i)$$

Thus,

$$\begin{aligned}\alpha_{i,K}(p|z_i) &= P_p\{Z_K \geq u_K, l_{K-1} < Z_{K-1}, \dots, l_{i+1} < Z_{i+1}|Z_i = z_i\} \\ &= P_p\left\{\frac{Z_{i,K} + Z_i\sqrt{I_i}}{\sqrt{I_K}} \geq u_K, l_{K-1} < \frac{Z_{i,K-1} + Z_i\sqrt{I_i}}{\sqrt{I_{K-1}}}, \dots, l_{i+1}\right. \\ &\left. < \frac{Z_{i,i+1} + Z_i\sqrt{I_i}}{\sqrt{I_{i+1}}}|Z_i = z_i\right\} \\ &= P_p\{Z_{i,K} \geq u_K\sqrt{I_K} - Z_i\sqrt{I_i}, Z_{i,K-1} \\ &> l_{K-1}\sqrt{I_{K-1}} - Z_i\sqrt{I_i}, \dots, Z_{i,i+1} > l_{i+1}\sqrt{I_{i+1}} - Z_i\sqrt{I_i}|Z_i = z_i\} \\ &= P_p\{Z_{i,K} \geq u_K\sqrt{I_K} - z_i\sqrt{I_i}, Z_{i,K-1} > l_{K-1}\sqrt{I_{K-1}} - z_i\sqrt{I_i}, \dots, Z_{i,i+1} \\ &> l_{i+1}\sqrt{I_{i+1}} - z_i\sqrt{I_i}\}\end{aligned}$$

Let $u_K^*(z_i) = u_K\sqrt{I_K} - z_i\sqrt{I_i}, l_m^*(z_i) = l_m\sqrt{I_m} - z_i\sqrt{I_i}, m = 1, \dots, K-1.$ Then,

$$\alpha_{i,K}(p|z_i) = P_p\{Z_{i,K} \geq u_K^*(z_i), Z_{i,K-1} > l_{K-1}^*(z_i), \dots, Z_{i,i+1} > l_{i+1}^*(z_i)\},$$

where $Z_{i,i+1}, \dots, Z_{i,K}$ jointly follow multivariate normal distribution with mean vector $\boldsymbol{\mu} = \{\theta(I_{i+1} - I_i), \theta(I_{i+2} - I_i), \dots, \theta(I_K - I_i)\}$ and a covariance matrix

$$\Sigma_{(K-i)\times(K-i)} = \begin{bmatrix} (I_{i+1} - I_i) & (I_{i+1} - I_i) & (I_{i+1} - I_i) & \cdots & (I_{i+1} - I_i) \\ (I_{i+1} - I_i) & (I_{i+2} - I_i) & (I_{i+2} - I_i) & \cdots & (I_{i+2} - I_i) \\ (I_{i+1} - I_i) & (I_{i+2} - I_i) & (I_{i+3} - I_i) & \cdots & (I_{i+3} - I_i) \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ (I_{i+1} - I_i) & (I_{i+2} - I_i) & (I_{i+3} - I_i) & \dots & (I_K - I_i) \end{bmatrix}$$

$\alpha_{i,K}(p|z_i)$ can be calculated by the function '`pmvnorm`' from R package '`mvtnorm`' [5].

Section 3 Remarks

(1) Calculate multiple integrals

The integrands of the multiple integrals in Section 1.3 and (2) of Section 2 are complex functions related to normal density functions which are not so straightforward to be integrated. For some software, multiple integrals are approximated by numeric integration carried out via Simpson's rule, a special case of Newton-Cotes formulas. R package "`mvtnorm`" provides function "`pmvnorm`" to compute the distribution function of the multivariate normal distribution with arbitrary limits and correlation matrices. `pmvnorm` performs

certain transformations on integral variables after which the multiple integral is restated as the product of several univariate integrals. With the function `pmvnorm`, the computation of asymptotic test in this package can be implemented efficiently.

(2) Binding boundaries vs non-binding boundaries

With simultaneously generated upper and lower boundaries, the trial must be terminated once the test statistic crosses the latter so as not to inflate the type I error. In this circumstance, the lower boundaries are strictly binding, which may not be favorable for the purpose of monitoring. Thus, making the lower boundaries non-binding seems to be a better answer. To construct the non-binding lower boundaries, upper boundaries are first generated as if there is no lower boundary, and later, fixed while searching for the lower boundaries. The whole procedure is iterated until requirements for two types of errors are strictly met (for asymptotic test) or asymptotically satisfied (for exact test).

(3) Carry over the unspent type II error when generating boundaries

Due to the discreteness of binomial distribution, in exact test, the type I and type II error actually spent at analysis j may not approximate the designated amount $\alpha(t_j) - \alpha(t_{j-1})$ and $\beta(t_j) - \beta(t_{j-1})$, respectively. It is natural to consider carrying over the unused portion of errors at any stage to the subsequent stages when generating boundaries. Since we only have one upper boundary u_K for the last look, there is no need to carry over type I error.

In **Step 3** of Algorithm 2, for $k = 2, \dots, K-1$, $P_{p_1}\{Z_1 \le l_1\} + P_{p_1}\{Z_1 > l_1, Z_2 \le l_2\} + \cdots + P_{p_1}\{Z_1 > l_1, \cdots, Z_{k-1} > l_{k-1}, Z_k \le l_k\} \le \beta(t_k)$, where the unused portion of type II error at earlier stages can be accumulated and spent at stage k. In contrast, if the type II error is not carried over to the subsequent looks, the formula becomes $P_{p_1}\{Z_1 > l_1, \cdots, Z_{k-1} > l_{k-1}, Z_k \le l_k\} \le \beta(t_k) - \beta(t_{k-1})$. Due to the lower tolerance of type II error, the new formula can result in smaller futility boundaries before the last look, making it less likely to reject the alternative hypothesis even when H_1 is not that advantageous. Remember that, in this document, the lower boundaries are non-binding, so the overall type I error would not be influenced by not carrying over the type II error.

To probe into the impact of not carrying over the type II error on the lower boundaries, we set up a simulation study which mimics a three-stage group sequential design. For simplicity, only futility boundary is considered. Let the overall type II error $\beta = 0.2$ and the sequence of information fractions $\{t_1, t_2, t_3\} = \{0.3, 0.6, 1\}$. Consider two beta spending functions, that is, $\{\beta(t_1), \beta(t_2), \beta(t_3)\} = \{\frac{\beta}{3}, \frac{2\beta}{3}, \beta\}$ and $\{\beta(t_1), \beta(t_2), \beta(t_3)\} = \{0.3\beta, 0.5\beta, \beta\}$. Lower boundaries are derived with and without carrying over unspent type II error, with the maximum sample size n_K varying from 30 to 190 by 20 and the alternative p_1 varying from 0.35 to 0.75 by 0.05. As there is no unspent type II error before the first analysis, the lower boundary for the first look will stay unchanged with or without carrying over type II error. When carrying over type II error, about half of the lower boundaries at the second stage are one larger than those obtained without carrying over type II error. At the third stage, 66% of the lower boundaries obtained with carrying over type II error are one or two greater than their competitors. Since the gap is not too wide, it won't hurt to generate the lower boundaries with carried-over type II error while to some extent reducing the possibility of getting false positive results.

(4) Increase the maximum sample size to enhance power

Remember that the lower boundaries are generated with the upper boundary fixed to make them non-binding. However, having controlled the overall type I error, the actual power achieved in **Step 4** of both Algorithm 1 and 2 may be lower than the required level $1 - \beta$. Other things being equal, larger sample size generally yields higher power. Here, we choose to increase the maximum sample size to enhance the level of power.

(5) Adopt the maximum sample size obtained from Algorithm 1 for initialization of Algorithm 2

As suggested by [4], when generating boundaries for exact test, it is useful to start with the results from asymptotic test. In **Step 0** of Algorithm 2, the two probabilities implicitly use the maximum sample size n_K. If n_K is not specified beforehand, we must solve two inequations with respect to both the upper boundary u_K and n_K, which requires massive computation. In fact, the sample sizes obtained from asymptotic test ought to be close to those from exact test.

Thus, it makes sense to adopt the n_K obtained from asymptotic test as an initial value for exact test.

(6) Asymptotic test vs exact test

Taking advantage of the smoothness of normal distribution, the overall two types of errors as well as type II error spent at each analysis can be exactly achieved by implementing the asymptotic test. However, with a small sample size, the asymptotic normal distribution may not be a good representative of the actual distribution of binary population. The way to estimate the sample variance $var(\hat{\theta}_k)$ is another source of imprecision.

Due to the discreteness of binomial distribution, the type II error spent at each analysis can only approximate the desired amount, where the strategy of carrying over the unspent type II error may be necessary. The saw-toothed power function also results in more than one possible choice of sample sizes. Nevertheless, the associated test statistic is essentially the count of responses, which is not influenced by the estimation of sample variance. Further, exact test precisely depicts the actual distribution of binary population even with a very small sample size.

Having weighed the pros and cons of the two tests, we decided to provide functions for both strategies.

Section 4 Quick start

Let's start with the calculation of sample sizes and boundaries under the asymptotic test.

If not otherwise restated, the setup of the trial is as follows: sequence of information fractions $I = \{0.2,0.4,0.6,0.8,0.99\}$; desired overall type II error rate $\beta = 0.2$; the sequence of portions of type II error spent at each analysis {0.1,0.2,0.3,0.3,0.2} or equally the values of beta spending function at the timing for each analysis $\beta(t_k) = \{0.1,0.3,0.6,0.9,1.1\}$; desired overall type I error rate $\alpha = 0.05$; response rate under null hypothesis $p_0 = 0.3$; response rate under alternative hypothesis $p_1 = 0.5$; number of planned analyses $K = 4.6$; the maximum acceptable difference between the desired type II error spending and the actual type II error spending while computing lower bounds under asymptotic test $tol = 10^{-6}$, hereafter, we refer to tol as tolerance level.

Firstly, user should define some variables to set up the design as follows:

```
> library(BinGSD)
> I=c(0.2,0.4,0.6,0.8,0.99)
> beta=0.2
> betaspend=c(0.1,0.2,0.3,0.3,0.2)
> alpha=0.05
> p_0=0.3
> p_1=0.5
> K=4.6
> tol=1e-6
```

To obtain the sample sizes and associated boundaries under asymptotic test, call t -he function `asymdesign`:

```
> tt1=asymdesign(I,beta,betaspend,alpha,p_0,p_1,K,tol)
Warning messages:
1: In check.asymdesign(I, beta, betaspend, alpha, p_0, p_1, K, tol) :
  I will be standardized so that the last element is 1.
2: In check.asymdesign(I, beta, betaspend, alpha, p_0, p_1, K, tol) :
  betaspend will be standardized so that the total is 1.
```

Note that the last element of I is 0.99 but not 1. In this case, function `asymdesign` will automatically divide I by 0.99 to make its last element equal 1. Since the sum of desired type II error spending proportions exceeds 1, `asymdesign` will also standardize the vector {0.1,0.2,0.3,0.3,0.2}. While executing the R commands, $K = 4.6$ will also be rounded up to 5.

`tt1` is an object of class `asymdesign`. It is of interest to find some important outputs such as sample sizes, boundaries, actual type I and type II error spending:

```
> tt1$n.I     #sample sizes required for each analysis
[1]  9 18 27 36 44
> tt1$lowerbounds    #lower bounds for each analysis
[1] -0.96146695 -0.08607206  0.61570293  1.12238155  1.64485363
> tt1$u_K        #the last and only upper bound
[1] 1.644854
> tt1$probhi    #the actual type I error
[1] 0.04290043
> tt1$problow       #the actual type II error for each analysis
[1] 0.01532964 0.02969010 0.04437652 0.04436548 0.06041841
> tt1$power    #the power for the group sequential test
[1] 0.8058198
```

For design under exact test, call the function `exactdesign`. The only input for `exactdesign` is the output from asymptotic.

```
> tt2=exactdesign(tt1) #design based on exact test
```

`tt2` is an object of class `exactdesign`. User could also find some important numbers via the commands below:

```
> tt2$n.I      #sample sizes required for each analysis
[1]  9 18 27 36 44
> tt2$lowerbounds   #lower bounds for each analysis
[1]  0  5  9 14 19
> tt2$u_K       #the last and only upper bound
[1] 19
> tt2$probhi     #the actual type I error
[1] 0.0360286
> tt2$problow    #the actual type II error for each analysis
[1] 0.001953125 0.046669006 0.032415666 0.063932401 0.044413624
> tt2$power    #the power for the group sequential test
[1] 0.8106162
```

Given a design, whether by inputting from keyboard or calling functions in `BinGSD`, our package enables the calculation of boundary crossing probabilities. There are two functions tailored for two kinds of tests: `asymprob`; `exactprob`.

Obtain boundary crossing probabilities of the user-defined designs:

```
> tt3=asymprob(K=5,p_0=0.4,p_1=c(0.5,0.6,0.7,0.8),n.I=c(15,20,25,30,3
+ 5),u_K=1.65, lowerbounds=c(-1.2,-0.5,0.2,0.8,1.65))

> tt4=exactprob(K=5,p_0=0.4,p_1=c(0.5,0.6,0.7,0.8),n.I=c(15,20,25,30,3
+ 5),u_K=15, lowerbounds=c(3,5,10,12,15))
```

For `asymprob` and `exactprob`, `p_1` is a mandatory input. User is assumed to input all other arguments except for d to define a design. The resulting `tt3` and `tt4` are two objects of class `asymprob` and `exactprob`, respectively. The probabilities of crossing the lower bounds under `p_0` and `p_1` are contained in a matrix named `problow`. `probhi` contains the upper bound crossing probabilities.

```
> tt3$problow
      p             1             2             3             4             5         Total
[1,] 0.4 1.150697e-01 1.993602e-01 2.710091e-01 2.087000e-01 1.569305e-01 9.510694e-01
[2,] 0.5 2.415697e-02 6.058913e-02 1.327723e-01 1.760468e-01 2.884341e-01 6.819993e-01
[3,] 0.6 2.708428e-03 8.042296e-03 2.391436e-02 4.550975e-02 1.442661e-01 2.244409e-01
[4,] 0.7 9.368515e-05 2.570261e-04 8.447077e-04 1.878732e-03 1.041188e-02 1.348603e-02
[5,] 0.8 1.958134e-07 2.899753e-07 6.711291e-07 1.127068e-06 8.741561e-06 1.102555e-05
> tt3$probhi
      p 1 2 3 4          5
[1,] 0.4 0 0 0 0 0.04893055
[2,] 0.5 0 0 0 0 0.31800074
[3,] 0.6 0 0 0 0 0.77555909
[4,] 0.7 0 0 0 0 0.98651397
[5,] 0.8 0 0 0 0 0.99998897
```

Obtain boundary crossing probabilities of the design defined by `tt1` or `tt2`:

```
> tt3=asymprob(p_1=c(0.4,0.5,0.6,0.7,0.8,0.9),d=tt1)
> tt4=exactprob(p_1=c(0.4,0.5,0.6,0.7,0.8,0.9),d=tt2)
> tt4$problow
      p             1             2             3             4             5         Total
[1,] 0.3 0.040353607 4.950472e-01 2.171319e-01 1.641748e-01 4.726390e-02 9.639714e-01
[2,] 0.4 0.010077696 1.996819e-01 1.368397e-01 2.006824e-01 1.110455e-01 6.583272e-01
[3,] 0.5 0.001953125 4.666901e-02 3.241567e-02 6.393240e-02 4.441362e-02 1.893838e-01
```

```
[4,] 0.6 0.000262144 5.614867e-03 2.698102e-03 5.130427e-03 3.144015e-03 1.684955e-02
[5,] 0.7 0.000019683 2.637614e-04 5.633442e-05 7.061388e-05 2.598474e-05 4.363774e-04
[6,] 0.8 0.000000512 2.475811e-06 1.196444e-07 5.629186e-08 7.094995e-09 3.170842e-06
[7,] 0.9 0.000000001 5.182848e-10 1.259926e-12 6.552126e-14 8.363796e-16 1.519611e-09
> tt4$probhi
      p 1 2 3 4         5
[1,] 0.3 0 0 0 0 0.0360286
[2,] 0.4 0 0 0 0 0.3416728
[3,] 0.5 0 0 0 0 0.8106162
[4,] 0.6 0 0 0 0 0.9831504
[5,] 0.7 0 0 0 0 0.9995636
[6,] 0.8 0 0 0 0 0.9999968
[7,] 0.9 0 0 0 0 1.0000000
```

For group-sequential test with K planned analyses, at most $K-1$ interim analyses can be done. Once collected some data, user is able to know the conditional probability that reject the null hypothesis at the final stage given the value of testing statistic at interim analysis. For function `asymcp`, argument `d` can be an object from either class `asymdesign` or class `asymprob`. For function `exactcp`, argument `d` can be an object from either class `exactdesign` or class `exactprob`.

Get the conditional power under asymptotic test:

```
> asymcp(tt1,p_1=c(0.4,0.5,0.6,0.7,0.8,0.9),1,2)$cp
      p        cp
[1,] 0.3 0.1947882
[2,] 0.4 0.6322987
[3,] 0.5 0.9304883
[4,] 0.6 0.9969857
[5,] 0.7 0.9999910
[6,] 0.8 1.0000000
[7,] 0.9 1.0000000
> asymcp(tt3,p_1=c(0.4,0.5,0.6,0.7,0.8,0.9),3,2.2) $cp
      p        cp
[1,] 0.3 0.5482361
[2,] 0.4 0.8318049
[3,] 0.5 0.9613503
[4,] 0.6 0.9958245
[5,] 0.7 0.9998914
[6,] 0.8 0.9999999
[7,] 0.9 1.0000000
```

To know the conditional power under exact test:

```
> exactcp(tt2,p_1=c(0.4,0.5,0.6,0.7,0.8,0.9),1,2) $cp
      p          cp
[1,] 0.3 0.009793508
[2,] 0.4 0.130988862
[3,] 0.5 0.487896752
[4,] 0.6 0.833918068
[5,] 0.7 0.969182514
[6,] 0.8 0.996833912
[7,] 0.9 0.999935684
> exactcp(tt4,p_1=c(0.4,0.5,0.6,0.7,0.8,0.9),3,10) $cp
      p         cp
[1,] 0.3 0.02696603
[2,] 0.4 0.14146984
```

```
[3,] 0.5 0.38434601
[4,] 0.6 0.67596567
[5,] 0.7 0.88788043
[6,] 0.8 0.97948791
[7,] 0.9 0.99910469
```

We hope this document is helpful for user to get started with '`BinGSD`'. Please reach Lei at slimewanglei@163.com to report bugs and share your experience using our package.

Reference

[1] Cytel Inc. East Version 6.4.1 Manual. 2017.

[2] SAS Institute Inc. SAS/STAT 15.1 User's Guide. 2018

[3] Keaven M. Anderson, Dan (Jennifer) Sun, Zhongxin (John) Zhang. gsDesign: An R Package for Designing Group Sequential Clinical Trials. R package version 3.0-1.

[4] Christopher Jennison, Bruce W. Turnbull. Group Sequential Methods with Applications to Clinical Trials. Chapman and Hall/CRC, Boca Raton, FL, 2000.

[5] Alan Genz et al. (2018). mvtnorm: Multivariate Normal and t Distributions. R package version 1.0-11.