MASTERARBEIT

An algorithm for the calculation of exact confidence intervals for adaptive group sequential trials

Ausgeführt am Institut für
Medizinische Statistik
der Medizinischen Universität Wien

unter der Anleitung von

Ao. Univ.-Prof. Mag. Dr. Werner Brannath

eingereicht an der Technischen Universität Wien
Fakultät für Informatik

von

Niklas Hack
9901424
Breitenfurterstrasse 380A/39
1230 Wien

Wien, 13.12.2007

Contents

1 Introduction

During the last years much research was spent on making mid-course corrections to the sample size of a clinical trial while the overall type I error rate of the test was preserved. Adaptive or flexible designs for clinical trials are attractive to clinical scientists and researchers since they provide a method to add flexibility to the frequentist paradigm. An important feature of adaptive designs is that the precise adaptation rule needs not to be pre-planed.

The first methods that allow for full flexibility were suggested by Bauer (1989), Bauer and Köhne (1994), Proschan and Hunsberger (1995). The methods were further developed by Cui, Hung and Wang (1999), Lehmacher and Wassmer (1999), Shen and Fisher (1999), Denne (2001), Müller and Schäfer (2001, 2004), Brannath et al. (2002) and Hartung and Knapp (2003).

Müller and Schäfer (2001, 2004) presented the most general way to make adaptive changes to an on-going group sequential clinical trial while preserving the overall type I error rate. Their method allows to make data dependent changes to the sample size, the spending function and the number and spacing of interim looks at one or more time points. Adaptations can depend on the observed data up to the interim analysis and if no adaptation is performed the originally planned group sequential analysis can be applied. Only in the case of adaptations a modified test statistic based on the conditional error rate has to be performed.

One of the limits of the Müller and Schäfer method is that they do not give details on the computation of confidence intervals, point estimates and p-values for adaptive group sequential clinical trials. This limits the applicability of the method to real clinical trials. According to ICH E9 guideline testing efficacy should be accompanied by unbiased estimates and confidence intervals.

In recent years there have been several approaches to calculate point estimates and confidence intervals following an adaptive change. An overview of point and interval estimation is given in Brannath et al. (2006). For instance Brannath, Posch and Bauer (2002) proposed the recursive combination tests which have similar flexibility as the Müller and Schäfer (2001) method, while also providing p-values and exact confidence intervals. However, these tests were not primarily intended for group sequential trials and are difficult to apply to group sequential designs. Lehmacher and Wassmer (1999) extended Jennison and Turnbulls (1989) method of repeated confidence intervals to adaptive designs, based on the inverse normal method. The method of Lehmacher and Wassmer (1999) allows to perform data driven sample size adaptations in a classic group sequential design, but does not allow changes of the spending function or the number of interim analyses.

Metha, Bauer, Posch and Brannath (2006) proposed an approach for the calculation of repeated confidence intervals for adaptive group sequential trials. The Müller and Schäfer (2001) method is applied to the dual tests derived from the repeated confidence intervals (RCI) of Jennison and Turnbull (1989). However, this method can only provide conservative coverage of the efficacy parameter δ. Brannath, Mehta and Posch (2007) extended the stage-wise adjusted confidence intervals of Tsiatis, Rosner and Mehta (1984) to adaptive designs. Stage-wise adjusted confidence intervals provide exact coverage for classic group sequential designs. In the case of design adaptations it cannot be guaranteed that the stage-wise adjusted confidence interval provides exact coverage in general. The reason is that the dual adjusted rejection regions may not be nested and thus the confidence regions may not be an interval. Extending the confidence regions to an interval will produce a conservative lower confidence bound. Their method shares an essential feature of Müller and Schäfer adaptive test: If no adaptation is performed a classical stage-wise adjusted confidence interval is computed. Only in the case of adaptations a modified construction principle is performed. Furthermore the computed confidence intervals and p-values are usually consistent, such that the Müller and Schäfer method rejects the null hypothesis $H_0 : \delta \leq 0$ if and only if the corresponding confidence interval excludes the parameter δ. In rare and very special cases it may occur that the test rejects, however, the confidence interval accepts. Brannath et al. (2007) have shown by simulations that the coverage probability of the confidence intervals is - from a practical point of view - exact.

This work describes a new algorithm for the calculation of the stage-wise adjusted one-sided confidence interval of Brannath, Mehta and Posch (2007). The presented algorithm is implemented in C and works via a specific bisection search method. This work is confined to one-sided confidence intervals.

In chapter 2 we introduce the Müller and Schäfer (2001) method and show how to apply it to the dual tests of confidence intervals. In chapter 3 we discuss repeated p-values and p-values based on the stage-wise ordering. The repeated confidence intervals are discussed in chapter 4. In chapter 5 we discuss stage-wise adjusted confidence intervals, at first the classical stage-wise adjusted confidence bounds and afterwards the adaptive version. Chapter

6 describes the algorithm for the adaptive confidence intervals based on the stage-wise ordering in Brannath, Mehta and Posch (2007). In chapter 7 we show how to integrate the C-source code into R and in chapter 8 we illustrate the implemented algorithm by a simulation study done with R. The C program is given in the Appendix A.

2 Review of adaptive and sequential designs

2.1 History of sequential designs

The investigation of sequential experiments started in the 17^{th} and 18^{th} century with the work of Huyghens, James Bernoulli, DeMoivre, Laplace and others who studied gambling systems. In the late 1920s the first application of sequential procedures started in the area of statistical quality control in the manufacturing production. In this context Dodge and Roming (1929) defined the first two-stage acceptance sampling plan. The test consisted of six parameters and was inducted to test components and classify them as effective or defective. The two parameters n_1, n_2 defined the sample size of stage 1 and stage 2, respectively. The critical parameters to decide if the components were defective or effective were defined by c_1, c_2 (acceptance values) and d_1, d_2 (rejection values) and were due to the condition that $d_1 > c_1 + 1$ and $d_2 = c_2 + 1$. In the first stage of the two-stage acceptance sampling plan, they took an initial sample size of n_1 and if the n_1 samples contained c_1 or less defectives the lot was accepted, but if d_1 or more defectives were detected the lot was rejected. If the number of defectives was between c_1 and d_1, the decision was displaced until the complete sample size n_2 was tested. In the secondary trial the number of defectives was compared to c_2 and d_2 and if the items contained c_2 or fewer defectives the whole production was accepted, but if d_2 or more defectives were detected the production was rejected.

This approach was the basic module for the development of multi-stage or multi sampling plans (Bartky 1943). The idea was that a two-stage plan can easily be generalized to a k-stage plan.

Only a few years later the theory of sequential analyses was strongly affected by the work of Abraham Wald (1948) and especially by his sequential probability ratio test (SPRT). The test considers a random number of observations with known distribution $f_{(x;\theta)}$ and unknown parameter θ. The null hypothesis $H_0 : \theta = \theta_0$ is tested against an alternative $H_A : \theta = \theta_A$. Thereby H_0 should have at most α and H_A at most β as probability of error. The SPRT is defined by the likelihood ratio test statistic and the rejection boundaries (a, b). The boundaries can be chosen such that the probability of a type I error (select H_A when H_0 is true) and a type II error (select H_0 when H_A is true) are nearly equal to the predefined α and β. The critical upper boundary for rejection can be calculated by the equation $A = \frac{1-\beta}{\alpha}$ and the lower boundary by $B = \frac{\beta}{1-\alpha}$. The test statistic has the following decision rules: If the likelihood ratio is bigger than or equal to A, reject the null hypothesis H_0, if the likelihood ratio is smaller or equal to B accept the null hypothesis H_0, but if the test statistic is smaller than A and bigger than B, continue with the observations. Under H_0 and H_A this procedure leads to lower expected sample sizes than the fixed sample tests.

Only one year later Wald and Wolfowitz (1948) proved that the SPRT not exceeds α and β and has the smallest possible expected sample size or "average sample number" (ASN) when either H_0 or H_A is true. One of the main disadvantages of the test is that the sample size is not bounded and so it can lead to a large variance of the sample size.

The early multi-stage or "group sequential" design was primary developed for industrial acceptance tests with a binary response. Later the development was focused on two-stage and three-stage procedures for a normal response: see, for example, Armitage and Schneiderman (1958), Schneiderman (1961), Dunnett (1961), Roseberry and Gehan (1964) and Armitage, McPherson and Rowe (1969). In this early work repeated numerical integration was used to calculate operating characteristics and ASN curves of multi-stage procedures for normal data. This method represents the basic tool to construct many group sequential tests. It also plays a decisive role in the calculation of significance levels and confidence intervals following a group sequential test. In the 1950's Armitage and Bross considered the use of sequential methods in the medical field. Elfring and Schultz (1973) presented a method for the comparison of two treatments with binary response and were the first who used the term "group sequential design".

One of the major impetus for group sequential methods came from Pocock (1977). He defined the first exact approach for the implementation of group sequential designs attending type I error and power requirements. One of his greatest achievements was to show that the nominal significance level of repeated significant tests for a normal response can also be used for a variety of other responses and situations, like e.g. normal with unknown variance, binomial or exponential.

Only two years later O'Brien and Fleming (1979) presented a different class of group sequential tests based on an adaptation of a truncated SPRT. These tests had conservative stopping boundaries at very early analyses, which turned out to be very appealing in practice. Furthermore, there was still the problem that in clinical trials the group size is in the majority of cases unequal. Slud and Wei (1982) and Lan and DeMets (1983) showed that group sequential methods can also be employed when group sizes are unequal and even unpredictable.

After the investigation of the two-sided group sequential tests of Pocock (1977) and of O'Brien and Fleming (1979) the one-sided tests followed very soon. DeMets and Ware (1980 and 1982) adapted the tests from Pocock and O'Brien and Fleming to the one-sided hypothesis. Jennison (1987) and Eales and Jennison (1992) reduced the expected sample size by searching for optimal group sequential one-sided tests.

With the work of Siegmund (1978, 1985), Tsiatis, Rosner and Metha (1984) and Kim and DeMets (1987) it is now possible to calculate interval estimates for group sequential tests. Furthermore, Fairbanks and Madsen (1982) and Madsen and Fairbanks (1983) proposed a method to calculate p-values and Whitehead (1986) presented a method to calculate point estimates for group sequential tests. The method of Jennison and Turnbull (1984, 1989) allows the construction of repeated confidence intervals for the parameter of interest.

2.2 Group sequential designs

In this thesis we restrict attention to one-sided group sequential designs without futility bounds. We consider a group sequential test (see for example, Jennison and Turnball (2000)) for a comparative study of an experimental treatment E to a control treatment C, with a total of N normally distributed observations $X_{il}, i = E$ or $C, l = 1, 2, \ldots, N/2$, with known variance σ^2. Let μ_E and μ_C denote the means based on a treatment E and a control C group and $\delta = \mu_E - \mu_C$ the difference of the population means. We focus on group sequential tests of the hypothesis

$$H_0 : \delta \leq 0$$

against the one-sided alternative $\delta > 0$. The trial is performed in K sequential stages after observing the cumulative responses for $n_1, n_2, \ldots, n_K = N$ subjects. At stage j the data are summarized by the Wald statistics

$$Z_j = \hat{\delta}_j \sqrt{I_j}, \qquad j = 1, \ldots, K$$

where $\hat{\delta}_j$ is the maximum likelihood estimate of δ and $I_j \approx \left[se\left(\hat{\delta}_j\right)\right]^{-2} = n_j/\left(4\sigma^2\right)$ is the estimate of the Fisher information. We calculate sequentially for every interim analysis the Wald statistic $\{Z_1, Z_2, \ldots, Z_K\}$. This statistic has the following known distributional properties: (Jennison and Turnbull, 2000, p.49)

$$(1)\, (Z_1, Z_2, \ldots, Z_K) \text{ is multivariate normal,}$$

$$(2)\, E\left(Z_j\right) = \delta\sqrt{I_j}, \qquad j = 1, 2, \ldots, K,$$

$$(3)\, Cov\left(Z_{j_1}, Z_{j_2}\right) = \sqrt{I_{j_1}/I_{j_2}}, \qquad 1 \leq j_1 \leq j_2 \leq K$$

This implies that $\{Z_1, Z_2, \ldots, Z_K\}$ is a Markov chain of the first order. The Markov chain property means that the conditional distribution of Z_j, conditional on the past history $(Z_{j-1}, Z_{j-2}, \ldots, Z_1)$, depends only on the previous stage Z_{j-1}. Using this property one can generate stopping boundaries $\{b_j, j = 1, 2, \ldots, K\}$ for testing the null hypothesis

$$H_0 : \delta \leq 0.$$

in such a way that under H_0

$$P_0\left(\bigcup_{j=1}^{K} Z_j \geq b_j\right) \leq \alpha$$

whereby this probability equals α if $\delta = 0$. The α-spending function can be used to establish the boundaries $b_1, b_2, \ldots, b_K$ for each interim monitoring point, given the overall α (see next section).

The trial stops at look j when the observed Wald statistic z_j is larger than the rejection boundary b_j. We denote by T the random variable which gives the stage where the trial stops.

2.3 Spending function approach

In clinical trials accruing data are monitored periodically for safety and efficacy, resulting in several looks at the interim data. In practice it is often difficult to perfectly meet the anticipated sample size for the respective interim analyses. Very often data is missing at the time-point of the pre-planed inspection of the data or the recruitment rate of the patients is lower than expected. The α-spending approach of Lan and DeMets (1983) allows us to perform interim analyses at arbitrary information fractions $t_j = \frac{I_j}{I_K}, j = 1, \ldots, K$, that are random and need not to be specified in advance. However, in order to keep the type I error α, the information fractions for the interim analyses must be independent from the data observed so far.

An α-spending function $g(t), 0 \leq g(t) \leq \alpha$, at level α, is a strictly increasing function in the infromation fraction $t \in (0, 1)$, which satisfies $g(0) = 0$ and $g(1) = \alpha$. The value $g(t_j)$ specifies the cumulative type I error rate "spent" at the j-th interim analysis. Typically lower $g(t)$-values are established for early looks and higher $g(t)$-values for later looks.

The spending function $g(t)$ determines the critical boundary at the first interim analysis by the equation $P_0(Z_1 \geq b_1) = g(t_1)$. Solving this equation we get $b_1 = \Phi^{-1}(1-g(t_1))$, where Φ^{-1} is the inverse standard normal cumulative distribution function. At time-point $t_2 = \frac{I_2}{I_K}$ the condition for the critical boundary b_2 is the solution of

$$P_0(\{Z_1 < b_1\} \cap \{Z_2 \geq b_2\}) = g(t_2) - g(t_1)$$

Hence, $g(t_2)$ is the level of significance that is spend till time-point t_2. We continue with the calculation till the complete type I error rate α is exhausted. In detail, the critical boundaries $b_j, j = 1, \ldots, K$, are recursively defined by

$$P_0\left(\bigcap_{j=1}^{k-1}\{Z_j < b_j\} \cap \{Z_k \geq b_k\}\right) = g(t_k) - g(t_{k-1}).$$

Examples for the choice of g(t) are

$$g(t_k) = \alpha \cdot \ln(1 + (e-1)t_k) \qquad \text{(Pocock, 1977)},$$

$$g(t_k) = 4(1 - \Phi\left(\frac{\Phi^{-1}(1-\frac{\alpha}{4})}{\sqrt{t_k}}\right)) \qquad \text{(O'Brien and Fleming, 1979) and}$$

$$g(t_k) = \frac{1-e^{-\gamma t}}{1-e^{-\gamma}} \qquad \text{(γ-family of Hwang, Shih and DeCani, 1990).}$$

where Φ is the standard normal cumulative distribution function.

2.4 Müller and Schäfer method

Müller and Schäfer (2001) present a general method for the full integration of the concept of adaptive interim analyses (Bauer and Köhne, 1994) into group sequential testing. This method is the first allowing to change statistical design elements of a given group sequential design such as the α-spending function and the number of interim analyses, without effecting the type I error rate. The method is described by statistical decision functions and is based on the conditional rejection probability of a decision variable.

The conditional rejection probability gives the conditional probability to finally reject the null hypothesis given the interim data, assuming that the null hypothesis is true. To explain the method, consider as in the previous section the case of a comparative study of an experimental treatment E to a control treatment C with means μ_E and μ_C and common known variance σ^2. As before assume a group sequential trial with $H_0 : \delta \leq 0$ against the one-sided alternative $H_A : \delta > 0$ and a maximum of K stages. Let us assume that the trial continues until stage $L < K$ without any rejection, i.e., $z_j < b_j$ for all $j \leq L$, where z_j is the observed value of the Wald test statistic Z_j from stage j. Let us further assume that one decides to make data dependent changes to the study design at look L. Let z_i denote the observed value of Z_i for $i \leq L$, and let R denote the event that H_0 will be rejected at any future analyses $j = L+1, \ldots, K$. R can be written as the union of disjoint events

$$R = \bigcup_{i=L+1}^{K} R_i$$

where

$$R_i = \{Z_i \geq b_i \text{ and } Z_j < b_j \text{ for all } j < i$$

The conditional probability for H_0 of the event R given Z_j for $j \leq L$, is called conditional rejection probability. It can formally be written as

$$\epsilon(0) = P_0(R|Z_1 = z_1, \ldots, Z_L = z_L).$$

Because of the Markov property of $Z_j, j \leq k$, we have that $\epsilon(0)$ is a function of z_L only as long as $z_i \leq b_i$ for all $i \leq L$. We now plan a new group sequential design at level $\epsilon(0)$. This trial starts at stage L and is based on a patient cohort which is independent from the cohort of patients recruited up to look L. This trial can be seen as a new, independent "secondary" trial in which the sample size is initialized to zero and the type I error is equal to $\epsilon(0)$. The Wald z-statistics for the secondary trial are only based on the data observed after the stage of the adaptation L. We will distinguish the secondary trial from the original "primary" trial by labeling the stages, sample sizes, stopping boundaries and test statistics by the superscript "(2)". Assume that the secondary trial has a maximum number of $K^{(2)}$ stages, cumulated information numbers $I_j^{(2)}, j = 1, \ldots, K^{(2)}$ and rejection boundaries $b_j^{(2)}, j = 1, \ldots, K^{(2)}$. The boundaries for the secondary group sequential trial have to be chosen in such a way, that the resulting test procedure has type I error $\epsilon(0)$, i.e.,

$$\epsilon(0) = P_0\left(\bigcup_{j=L+1}^{k^{(2)}} \left\{Z_j^{(2)} \geq b_j^{(2)}\right\} |Z_1 = z_1, \ldots, Z_L = z_L\right)$$

Assume that the secondary trial terminates at look $T^{(2)} \leq K^{(2)}$ with the observed test-statistic $Z_{T^{(2)}}^{(2)} = z_{T^{(2)}}^{(2)}$. Now, the null hypothesis is rejected if and only (if $z_{T^{(2)}}^{(2)} \geq b_{T^{(2)}}^{(2)}$). Note that the conditional rejection probability is the only information which is carried over to the secondary trial.

2.4.1 Numerical example

For a better understanding we illustrate the approach of Müller and Schäfer (2001) by a numerical example. We start with a comparative study of an experimental treatment E to a control treatment C. The trial is planed as a four-look, one-sided group sequential design at level $\alpha = 0.025$. We initially want to test $H_0 : \delta \leq 0$ with 90% power to detect $\delta = 5$ with known standard deviation $\sigma = 15$. The stopping boundaries are derived from the γ-family proposed by Hwang, Shih and DeCani (1990) with $\gamma = -4$, see section 2.3. Such boundaries are frequently used in practice. The boundaries were calculated with the statistical software EAST and have the values $b_1 = 3.155, b_2 = 2.818, b_3 = 2.439$ and $b_4 = 2.014$. The initial maximum sample size is $N = 392$ (up rounded to achieve equal sample sizes per group).

Now suppose that at the first interim analysis, after $n_1 = 98$ subjects in total (both groups together) have been evaluated, the estimate of δ is $\hat{\delta}_1 = 3$ with the estimated standard deviation $\hat{\sigma}_1 = 20$ which gives $z_1 = 0.742$. Since the observed $\hat{\delta}_1$ is below the anticipated δ and $\hat{\sigma}_1$ is higher than assumed we decide to increase the sample size. Given the data the conditional rejection probability is $\epsilon(0) = 0.031$. As described above we set the significance level of the secondary trial equal $\epsilon(0)$ to control the type I error rate . The sample size is calculated on the bases of $\delta = 4$, which is the mean of the original δ and the interim estimate $\hat{\delta}_1 = 3$, with $\sigma = 20$ and a power of 90%. The secondary trial is designed as a four-stage O'Brien-Fleming group sequential design (γ-family with $\gamma = -4$). The required total sample size is $N^{(2)} = 1016$, as was calculated with EAST. The corresponding rejection boundaries are $b_1^{(2)} = 3.092, b_2^{(2)} = 2.747, b_3^{(2)} = 2.357$ and $b_4^{(2)} = 1.918$.

Suppose that the secondary trial stops at the third stage with a final estimate $\hat{\delta}_3^{(2)} = 4$ and $\hat{\sigma}_3^{(2)} = 20$. The standardized test statistic at this stage is $z_3^{(2)} = 2.76$ and hence that $z_3^{(2)} > b_3^{(2)}$ we stop the trial and reject H_0.

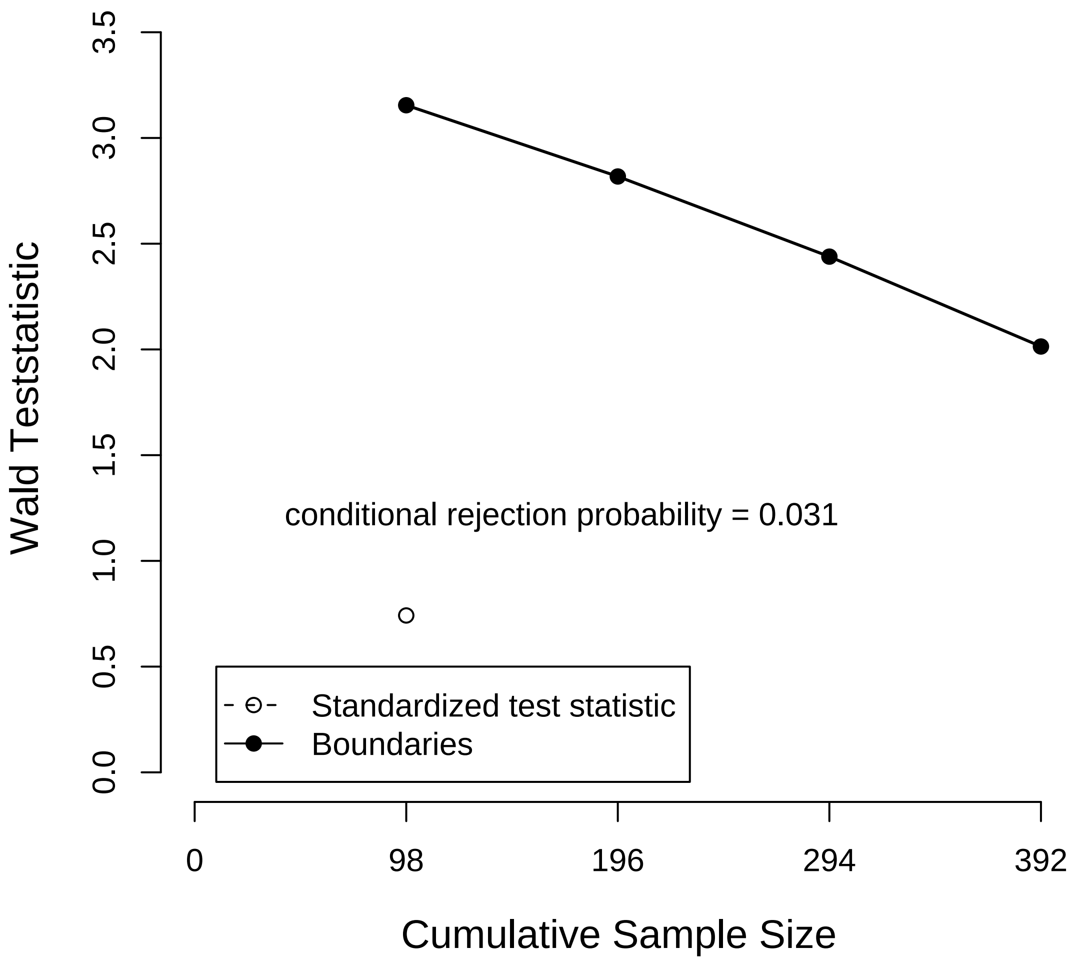

Figure 1: Primary trial (OBF design at level 0.025) from the example in section 2.4.1

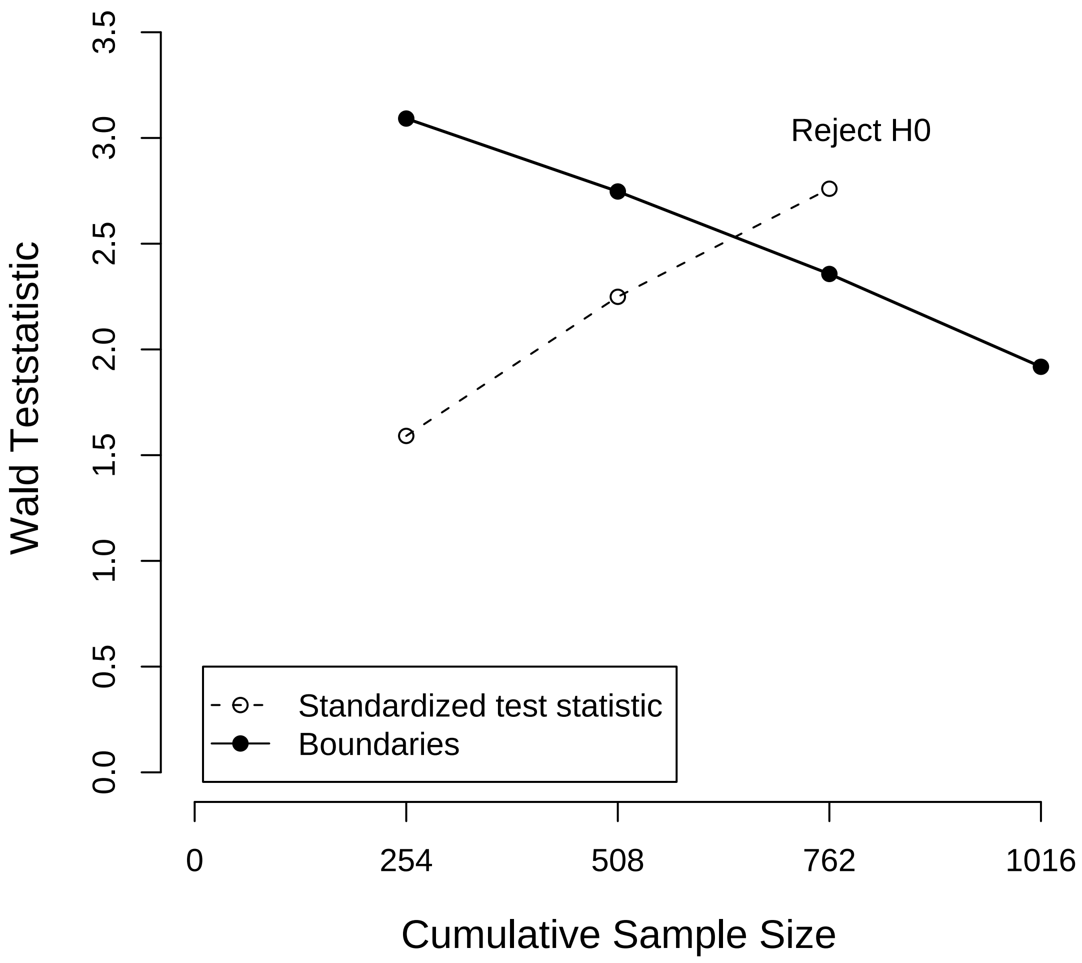

Figure 2: Secondary trial (OBF design at level 0.031) from the example in section 2.4.1

3 Overall p-values

An overall p-value can be defined via a family of nested hypotheses tests. We denote a family of hypotheses tests as nested, if the rejection of the level-u test in the family implies the rejection of all level-u' tests where $u' > u$. An overall p-value q can be defined as the minimum of the levels of the tests which reject H_0. In other words, we continue rejecting $H_0 : \delta \leq 0$ in a sequence of nested tests, with decreasing significant levels $0 < u < 1$, until we reach the level q, such that we cannot reject H_0. In the next chapters we are going to introduce the repeated p-value and the p-value based on the stage-wise ordering, which are both following this construction principle. Repeated p-values are flexible with respect to the stopping rule, but they are strictly conservative. P-values based on the stage-wise ordering are exact in the sense that the level is exhausted, but they can only be calculated at the stage where the trial stops according to the stopping rule.

3.1 Repeated p-values

For the repeated p-value the rejection boundaries of the primary trial and, in case of a design adaptation, also for the secondary trial, need to be specified via spending functions $g_u(t)$ and $g_u^{(2)}$ that generates boundaries $b_{k,u}$ and $b_{k,u}^{(2)}$ for all levels $0 < u < 1$ which are non-decreasing in u, or directly via a monotone family of boundaries $b_{k,u}$ and $b_{k,u}^{(2)}$.

3.1.1 No adaptive change

We start with the primary trial and choose a family of spending functions $g_u(t)$, $0 \leq u \leq 1$, such that $g_u(1) = u$ for all u. Now we calculate the test statistics at information times t_j and denote them by z_j. The critical boundaries, following from the spending function $g_u(t)$ at the stages $j = 1, \dots, K$, are denoted by $b_{j,u}$, where $b_{j,u}$ satisfies the usual level-u group sequential test requirement

$$P_0 \left(\bigcap_{j=1}^{K} \{ Z_j < b_{j,u} \} \right) = 1 - u.$$

In order to obtain nested rejection regions we must have $b_{j,u} < b_{j,u'}$ for all $0 \leq u' < u \leq 1$. This requires a specific assumption on the spending functions, see Mehta, Bauer, Posch and Brannath (2007). Now we can define the repeated p-value at stage k by

$$p_k = \inf\{u : z_k \geq b_{k,u}\} = \sup\{u : z_k < b_{k,u}\} \tag{1}$$

Note that $\bigcup_{j=1}^{K}\{p_j \leq u\} = \bigcup_{j=1}^{K}\{Z_j \geq b_{j,u}\}$ is the rejection rule of a group sequential test at level u. Hence, for any stopping rule T, $P(p_T \leq u) \leq P(\bigcup_{j=1}^{k}\{p_j \leq u\}) = u$.

3.1.2 Incorporating adaptive changes

Now let us assume that we perform some design adaptations at stage L. The conditional type I error rate for the test at level u is then given by

$$\epsilon_u = \begin{cases} 0 & \text{if } u \leq \alpha_L \\ P_0\left(\bigcup_{j=L+1}^{K}\{Z_j \geq b_{j,u}\} | Z_1 = z_1, \ldots, Z_L = z_L\right) & \text{if } u > \alpha_L \end{cases}$$

Let $p^{(2)}$ denote the repeated p-value of the secondary trial of the stage where the trial stops $T^{(2)}$, i.e.,

$$p^{(2)} = \inf(u : z^{(2)}_{T^{(2)}} \geq b^{(2)}_{T^{(2)},u})$$

where $b^{(2)}_{k,u}$ is from the monotone family of boundaries from the spending function for the secondary trial. Now the overall p-value, considering the data from the primary and secondary trial, with regard to the design adaptations at stage L is defined by

$$q = \inf\{u : p^{(2)} \leq \epsilon_u\} \tag{2}$$

In this equation ϵ_u is increasing in u (if all $b_{j,u}$'s are decreasing in u) and hence the corresponding adaptive level-u tests are nested. Therefore the p-value can be computed as the solution of the equation $p^{(2)} = \epsilon_u$.

Remark:

1. Repeated p-values can be defined at every interim look j of an adaptive secondary trial and not just at the look $T^{(2)}$ where the trial was terminated.
2. Repeated p-values produce conservative tests at levels $u \neq \alpha$.

We continue with the numerical example from section 2.4.1. The overall p-value q based on the secondary trial can now be calculated by (2) and is the unique solution of the equation $p^{(2)} = \epsilon_u$. In our example the overall p-value is $q = 0.0094$.

3.2 P-values based on the stage wise ordering

3.2.1 No adaptive change

We assume that the primary trial stops at look T. The stage wise ordering on the sample space (T, z_T) of the trial was proposed by Armitage (1957), Siegmund (1978), Fairbanks and Madsen (1982) and Tsiatis, Rosner and Mehta (1984). It considers a sample point (j, z_j) as more extreme than the sample point (k, z_k), if either $j < k$ or $j = k$ and $z_j \geq z_k$. This ordering can be used to define an overall p-value p for H_0 as

$$p = P_0 \left(\bigcup_{j=1}^{T-1} \{Z_j \geq b_j\} \cup \{Z_T \geq z_T\} \right) \tag{3}$$

which is the probability under H_0 to get a more extreme sample (in the sense of the stage wise ordering) than the one we have observed.

For a better understanding of this p-value let us define for $j = 1, 2, \ldots, K-1$,

$$\alpha_j = P_0 \left(\bigcup_{j=1}^{k} \{Z_j \geq b_j\} \right)$$

We further define for each $u \in [0, 1]$ the index $j_u = j$ in such a way that $\alpha_{j-1} < u \leq \alpha_j$ where $\alpha_0 = 0$ and $\alpha_K = 1$. α_j is the level "spent" in the primary trial at the j-th interim look.

Now we define the "threshold boundary" $b_{k,u}$ in such a way that it satisfies the relationship

$$P_0\left(\bigcup_{j=1}^{j_u-1}\{Z_j \geq b_j\} \cup \{Z_{j_u} \geq b_{j_u,u}\}\right) = u$$

One can show that $\{p \leq u\} = \bigcup_{j=1}^{j_u-1}\{Z_j \geq b_j\} \cup \{Z_{j_u} \geq b_{j_u,u}\}$.

3.2.2 Incorporating adaptive changes

In the case of a design adaptation at look L we compute the corresponding conditional error functions

$$\epsilon_u = \begin{cases} 0 & \text{if } u \leq \alpha_L \\ P_0\left(\bigcup_{j=1}^{k-1}\{Z_j \geq b_j\} \cup \{Z_k \geq b_{k,u}\} | Z_1 = z_1, \ldots, Z_L = z_L\right) & \\ & \text{if } \alpha_{k-1} < u \leq \alpha_k, k = L+1, \ldots, K \end{cases}$$

Let $p^{(2)}$ denote the stage-wise adjusted p-value of the secondary trial of the stage where the trial stops $T^{(2)}$, i.e.,

$$p^{(2)} = P_0\left(\bigcup_{j=1}^{T^{(2)}-1}\{Z_j^{(2)} \geq b_j^{(2)}\} \cup \{Z_{T^{(2)}}^{(2)} \geq z_{T^{(2)}}^{(2)}\}\right) \tag{4}$$

Now we can calculate the overall p-value by

$$q = \inf\{u : p^{(2)} \leq \epsilon_u\} = \sup\{u : p^{(2)} > \epsilon_u\} \tag{5}$$

where $p^{(2)}$ is the p-value of the secondary trial according to the stage-wise ordering of the secondary trial. Note that ϵ_u is monotonically increasing in u and hence the adaptive tests are nested and q can be evaluated as the root of $p^{(2)} = \epsilon_u$.

We continue with the numerical example from section 2.4.1. The p-value based on the stage-wise ordering can be calculated by (4) and is the unique solution of the equation $p^{(2)} = \epsilon_u$. In our example the stage-wise adjusted p-value is $q = 0.0076$.

4 Construction of one sided repeated confidence intervals

4.1 Confidence intervals

An $(1-\alpha)$ confidence interval gives an estimated range of parameter values for the unknown population parameter. The estimated range is calculated from a given set of sample data. Confidence intervals are usually calculated for a coverage probability of 95%, but also other coverage probabilities are possible. A coverage probability of 95% means that, with a probability of 95%, we receive an interval that contains the unknown parameter. The width of the confidence interval gives some idea about how uncertain the unknown parameter is. In the case of a lower one-sided confidence interval we only declare a value for the lower bound, and the upper bound is set to ∞. Confidence intervals are more informative than the result of a hypothesis test (where it is only decided to reject H_0 or don't reject H_0) since they provide a range of plausible values for the unknown parameter.

4.2 Classical repeated confidence bounds

We next review the classical Jennison and Turnbull (1989) one-sided confidence intervals for a given group sequential design, without any design modification. These repeated confidence intervals will be the starting point for adaptive repeated confidence intervals.

The repeated confidence interval is defined by a family of dual significance tests for the hypothesis $H_h : \delta \leq h$ versus $\delta > h$ for all $h \in (-\infty, \infty)$. The confidence interval includes all values of h where the shifted null hypothesis H_h is not rejected. First, we sequentially compute the shifted Wald statistics $Z_j(h) = Z_j - h\sqrt{I_j}, j = 1, 2, \ldots, K$, where I_j is the cumulated information until stage j. It is known that $Z_j(h)$ is $N(0,1)$-distributed under H_h. Now we apply the same group sequential design to all h. At stage j we reject H_h if $Z_j - h\sqrt{I_j} \geq b_j$, i.e., we reject all $h \leq \frac{Z_j - b_j}{\sqrt{I_j}}$. Hence, the lower confidence bound at each step $j = 1, 2, \ldots, K$ of the one-sided confidence interval is

$$\left(\underline{\delta}_j, \infty\right), j = 1, 2, \ldots, K, \qquad \text{with } \underline{\delta}_j = \frac{Z_j - b_j}{\sqrt{I_j}}$$

Furthermore, because

$$P_\delta\left(\bigcap_{j=1}^{K}\left\{\delta > \frac{Z_j - b_j}{\sqrt{I_j}}\right\}\right) = P_\delta\left(\bigcap_{j=1}^{K}\{\delta > \underline{\delta}_j\}\right) =$$

$$P_\delta\left(\bigcap_{j=1}^{K}\left\{Z_j - \delta\sqrt{I_j} < b_j\right\}\right) = 1 - \alpha \tag{6}$$

the interval $(\underline{\delta}_j, \infty)$ contains δ with probability greater than or equal to $1-\alpha$. Hence, the interval provides conservative coverage of δ at whatever stage the confidence bound is computed. This also implicates that the specified α-level is exhausted with the intersection with all $(\underline{\delta}_j, \infty)$. Hence, each single $(\underline{\delta}_j, \infty)$, e.g., the one at the stage the trial stops, is strictly conservative.

4.3 Adaptive repeated confidence bounds

In the case of a design adaptation we apply the Müller and Schäfer principle to all dual tests. Collecting all h where $H_h : \delta \leq h$ is accepted gives the $1-\alpha$ confidence interval. To obtain this confidence interval we shift the observed test statistic of the primary trial to

$$z_j(h) = z_j - h\sqrt{I_j}, \qquad j = 1, 2, \ldots, L,$$

and the test statistic observed in the secondary trial is shifted to

$$z_j^{(2)}(h) = z_j^{(2)} - h\sqrt{I_j^{(2)}}, \qquad j = 1, 2, \ldots, T^{(2)}.$$

Now, the conditional rejection probability can be calculated by

$$\epsilon(h) = P_0\left(\bigcup_{j=L+1}^{K}\{Z_j \geq b_j\} | Z_1 = z_1 - h\sqrt{I_1}, \ldots, Z_L = z_L - h\sqrt{I_L}\right).$$

Since the probability of crossing the boundaries b_j at some stage $j > L$ is decreasing with decreasing starting point $z_L - h\sqrt{I_L}$, we get that $\epsilon(h)$ is decreasing with increasing h. With the Müller and Schäfer (2001) principle we can define the family of dual tests for H_h with rejection rule

$$p^{(2)}(h) \leq \epsilon(h), \tag{7}$$

where $p^{(2)}(h)$ is a p-value of the secondary trial for the shifted test statistics $z_j^{(2)} - h\sqrt{I_j^{(2)}}$. To preserve the flexibility of the repeated confidence intervals we use the repeated p-value for $p^{(2)}(h)$. The rejection rule $p^{(2)}(h) \leq \epsilon(h)$ gives a level α-test for $H_h : \delta \leq h$. Note that $p^{(2)}(h)$ is increasing and $\epsilon(h)$ is decreasing in h. Applying (7) to all values of h gives the one-sided confidence interval $(\underline{\delta}, \infty)$ where $\underline{\delta}$ is the unique solution of $p^{(2)}(h) = \epsilon(h)$ in h.

We continue with the numerical example from section 2.4.1. For the computation of the confidence interval all information numbers I_j and $I_j^{(2)}$ are computed assuming that $\sigma = 20$. The repeated confidence interval is calculated at the stage $T^{(2)}$ where the trial stops and is the unique solution of $p^{(2)}(h) = \epsilon(h)$. In our example the repeated confidence interval is $(0.491, \infty)$.

4.4 Conservative point estimates

The lower bound of a one-sided confidence interval at level 0.5 is an point estimate for the true treatment effect δ. The median of such an estimate $\underline{\delta}_{0.5}$ is smaller or equal to the true effect and hence under-estimates the true δ. This implies that this point estimate is conservative. To obtain this level 0.5 lower confidence bound for δ we perform level 0.5 tests for all $H_h : \delta \leq h$. The confidence interval consists of all h where H_h is accepted.

4.4.1 No adaptive change

To obtain a conservative point estimate for the classic group sequential trial we first have to calculate rejection boundaries $b_{j,0.5}$ at level 0.5, e.g., by using a spending function $g_{0.5}(t)$ at level 0.5. Let us assume that the trial stops at stage T, then we calculate the classic repeated confidence bound by

$$\underline{\delta}_T = \tfrac{Z_T - b_{T,0.5}}{\sqrt{I_T}}$$

Now $\underline{\delta}_T$ is a conservative point estimate for the true effect δ.

4.4.2 Incorporating adaptive changes

Let us assume that we perform adaptive changes at stage L of the primary trial. Let us further assume that the secondary trial stops at stage $T^{(2)}$.

First we calculate the conditional rejection probability $\epsilon_{0.5}$ based on the new boundaries $b_{j,0.5}$ for $j = L+1, \ldots, K$. Then, as mentioned before, the z-statistic z_L at stage L is shifted to $z_L(h) = z_L - h\sqrt{I_L}$. The new boundaries $b^{(2)}_{j,0.5}$ for the secondary trial can be computed by using the spending function $g_{\epsilon_{0.5}(h)}(t)$ and the shifted z-statistics $z^{(2)}_j$ of the secondary trial $z^{(2)}_j(h) = z^{(2)}_j - h\sqrt{I^{(2)}_j}$. Now we perform a test for H_h and reject if $z^{(2)}_{T^{(2)}}(h) \geq b^{(2)}_{T^{(2)},0.5}(h)$. The conservative point estimate $\underline{\delta}_{0.5}$ is the value of h at which $z^{(2)}_{T^{(2)}}(h) = b^{(2)}_{T^{(2)},0.5}(h)$.

5 Stage-wise confidence intervals

The stage-wise adjusted confidence intervals proposed by Brannath, Mehta and Posch (2007) are less conservative than the repeated confidence intervals of Mehta, Bauer, Posch and Brannath (2006). The stage-wise adjusted confidence intervals also provide a less conservative point estimate for δ. The repeated confidence intervals have the advantage that they can be computed at any stage and are also valid if one deviates from the stopping rules of the primary and secondary trial. In contrast, the stage-wise confidence intervals can only be computed at the stage the trial stops according to the pre-specified stopping rule. Hence, with the stage-wise confidence intervals we cannot deviate from the pre-specified stopping rule.

5.1 Classic stage-wise confidence bound

The stage wise ordering can be used to define an overall p-value $p(h)$ for H_h as

$$p(h) = P_h\left(\bigcup_{j=1}^{T-1}\{Z_j \geq b_j\} \cup \{Z_T \geq z_T\}\right) \tag{8}$$

By definition $p(h)$ has an uniform distribution under H_h. Since $p(h)$ is monotone increasing in h, the equation $p(h) = \alpha$ has a unique solution. Thus we perform a level-α test for H_h, if we reject H_h in the case that $p(h) \leq \alpha$, and otherwise accept H_h.

It is difficult to apply the Müller and Schäfer (2001) principle directly to the rejection rule $p(h) \leq \alpha$. Therefore it is helpful to rewrite the rule $p(h) \leq \alpha$ as group sequential plan in terms of rejection boundaries for the Z_j's. To this end we define so called α-absorbing constants $\delta_1 \geq \delta_2 \geq \ldots \geq \delta_{K-1}$ such that, for $k = 1, 2, \ldots, K-1$,

$$P_{\delta_k}\left(\bigcup_{j=1}^{k}\{Z_j \geq b_j\}\right) = \alpha \tag{9}$$

Note that the probability to reject at some given stage k increases in the parameter δ and reaches α at δ_k. Hence, when testing H_{δ_k} via the stage-wise ordering the level is used up (absorbed) at stage k. We further define $\delta_0 = \infty$ and $\delta_K = 0$ so that we can find for every real value of h the unique index $k(h) = k$ such that $\delta_k \leq h < \delta_{k-1}$. For each such h we can define the "threshold boundary" $b_{k(h)}(h)$ by the identity

$$P_h \left(\bigcup_{j=1}^{k(h)-1} \{Z_j \geq b_j\} \cup \left\{ Z_{k(h)} \geq b_{k(h)}(h) \right\} \right) = \alpha \tag{10}$$

For given h we can determine $b_{k(h)}(h)$ by a root finding process. It can be shown that the rejection rule $p(h) \leq \alpha$ is equivalent to a group sequential design with a maximum of $k(h)$ stages. The boundaries of this group sequential design are b_j for $j \leq k(h) - 1$ and $b_{k(h)}(h)$ at stage $k(h)$, i.e.,

$$\{p(h) \leq \alpha\} = \bigcup_{j=1}^{k(h)-1} \{Z_j \geq b_j\} \cup \{Z_{k(h)} \geq b_{k(h)}(h)\},$$

where we put $\bigcup_{j=1}^{0} \{Z_j \geq b_j\} = \emptyset$. The function $b_{k(h)}(h)$ assures the type I error rate α.

Note that the p-value of $p(h)$ is equal to α if the observed value of $Z_{k(h)}$ equals $b_{k(h)}(h)$. This follows from (8) and (10). Note further that $p(h)$ is a monotone increasing function of h. It follows from identity (10) that if h increases from $\delta_K = -\infty$ to δ_{K-1}, then $b_{K-1}(h)$ increases from $-\infty$ to ∞. If $k < K$ and h increases from δ_{k-1} to δ_k then $b_k(h)$ increases from b_k to ∞.

5.2 Construction principle for the stage-wise confidence bound

Let us now assume that we want to perform some design adaptations at look L. Recall that we have to compute the conditional type I error rate

$$\epsilon(0) = P_0 \left(\bigcup_{j=L+1}^{K} \{Z_j \geq b_j\} \,|Z_1 = z_1, \ldots, Z_L = z_L \right)$$

As explained in chapter 2.4, whatever secondary trial we choose it must be at level $\epsilon(0)$.

In order to test $H_h : \delta \leq h$ at level α we apply the Müller and Schäfer (2001) principle for any given h by computing the conditional error function of the test for H_h

$$\epsilon(h) = P_h \left(\bigcup_{j=L+1}^{k(h)-1} \{Z_j \geq b_j\} \cup \left\{ Z_{k(h)} \geq b_{k(h)}(h) \right\} | Z_1 = z_1, \ldots, Z_L = z_L \right). \tag{11}$$

We assume that the secondary trial stops at look $T^{(2)}$. Then we compute the p-value according to the stage-wise ordering of the secondary trial as

$$p^{(2)}(h) = P_h \left(\bigcup_{j=1}^{T^{(2)}-1} \left\{ Z_j^{(2)} \geq b_j^{(2)} \right\} \cup \left\{ Z_{T^{(2)}}^{(2)} \geq z_{T^{(2)}}^{(2)} \right\} \right)$$

With this new p-value we can define the dual test in such a way that $H_h : \delta \leq h$ is rejected if and only if $p^{(2)}(h) \leq \epsilon(h)$.

With the above adaptive tests for H_h it is now possible to compute the lower confidence bound $\underline{\delta}$ in the case of an adaptive change at look L. We build the confidence set of all parameter values h that were accepted, i.e., $p^{(2)}(h) > \epsilon(h)$. We have the problem that $p^{(2)}(h) = \epsilon(h)$ may have more than one solution. The reason is the non-monotonicity of $\epsilon(h)$ (see Figure 3 and Brannath et al., 2007). At the present we are unable to identify precise conditions under which $\epsilon(h)$ is not increasing, or is increasing at a slower rate than $p^{(2)}(h)$. Thus we define $\underline{\delta}$ as the smallest solution of $p^{(2)}(h) = \epsilon(h)$ which gives a conservative lower confidence bound.

We continue with the numerical example from section 2.4.1. The stage-wise adjusted confidence interval can now be calculated and is the smallest solution of $p^{(2)}(h) = \epsilon(h)$. In our example the stage-wise adjusted confidence interval is $(0.786, \infty)$.

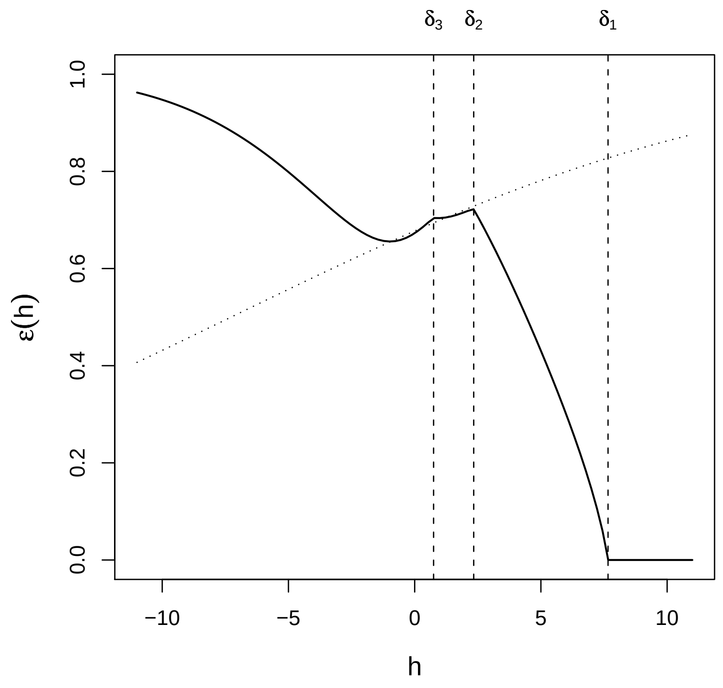

Figure 3: In Figure 3 we show an example for the nonmonotonicity of the conditional rejection probability $\epsilon(h)$ in h for the stage-wise adjusted confidence intervals. The trial is designed as a four-stage O'Brien-Fleming group sequential design with equally spaced looks at level $\alpha = 0.025$. The conditional rejection probability is computed at look 1 and the z-statistic observed at this look is $z_1 = 4$. The secondary trial is designed as a one stage group sequential design at level $\epsilon = 0.673$ with the upper boundary $b_1^{(2)} = -0.44792$ and the lower bound $a_1^{(2)} = -10$. The z-statistic at the end of the secondary trial is $z_1^{(2)} = -0.46$. The dotted line are the p-values $p^{(2)}(h)$ in h and the vertical dashed lines show the α-absorbing parameters $\delta_1 \geq \delta_2 \geq \delta_3$. We can now see that the p-value $p^{(2)}(h)$ crosses $\epsilon(h)$ more than one time and hence we cannot find an unique solution for $p^{(2)}(h) = \epsilon(h)$.

5.3 Median unbiased point estimates

To calculate the point estimate $\underline{\delta}_{0.5}$ based on the stage-wise ordering we calculate the stage-wise confidence interval at level 0.5. If the primary trial terminates without any adaptation at stage T, then $\underline{\delta}_{0.5}$ is the value of h that satisfies $p(h) = 0.5$ where $p(h)$ is calculated by (8).

Let us now assume that we perform a design adaptation at stage L of the primary trial. We first have to calculate new α-absorbing constants $\delta_{1,0.5} \geq \delta_{2,0.5}, \ldots, \delta_{K-1,0.5}$ for the level 0.5, i.e., such that

$$P_{\delta_k,0.5}\left(\bigcup_{j=1}^{k}\{Z_j \geq b_j\}\right) = 0.5, \qquad \text{k=1,2,...,K-1.}$$

We further define $\delta_{0,0.5} = \infty$ and $\delta_{K,0.5} = 0$. Now we calculate the "threshold boundary" $b_{k,0.5}(h)$ from the identity

$$P_h\left(\bigcup_{j=1}^{k-1}\{Z_j \geq b_j\} \cup \{Z_k \geq b_{k,0.5}(h)\}\right) = 0.5.$$

We assume that the secondary trial stops at look $T^{(2)}$. Then we compute the conditional rejection probability similar as in section 5.2 by

$$\epsilon_{0.5}(h) = P_h\left(\bigcup_{j=L+1}^{k-1}\{Z_j \geq b_j\} \cup \{Z_k \geq b_{k,0.5}(h)\} | Z_1 = z_1, \ldots, Z_L = z_L\right)$$

The median unbiased point estimate $\underline{\delta}_{0.5}$ is the smallest value of h such that $p^{(2)}(h) = \epsilon_{0.5}(h)$.

6 The algorithm

To construct the confidence interval $(\underline{\delta}, \infty)$ we first mention, that the conditional type I error rate can also be written as $\epsilon = A(h) + B(h)$ with

$$A(h) = P_h \left(\bigcup_{j=L+1}^{k(h)-1} \{Z_j \geq b_j\} \,|Z_1 = z_1, \ldots, Z_L = z_L \right)$$

and

$$B(h) = P_h \left(\bigcap_{j=L+1}^{k(h)-1} \{Z_j < b_j\} \cap \left\{ Z_{k(h)} \geq b_{k(h)}(h) \right\} |Z_1 = z_1, \ldots, Z_L = z_L \right)$$

It is known that Z_j is stochastically increasing in h, and so one can show that $A(h)$ is increasing in h and $B(h)$ is decreasing in h, for details see Brannath et al. (2007). The algorithm relies on these monotonicity properties of $A(h)$ and $B(h)$, and it has the ability to find the (smallest) solution of $p^{(2)}(h) = \epsilon(h)$.

The main idea of the algorithm is the following. We start with the partition $(-\infty, 0], (0, \delta_{K-1}], \ldots, (\delta_{L+1}, \delta_L]$ of the interval $(-\infty, \delta_L]$ which is known to contain the lower confidence bound $\underline{\delta}$. We try to identify the interval of the partition which contains $\underline{\delta}$. This may not be possible with the initial partition, however, by refining the partition we will finally be able to identify the interval containing $\underline{\delta}$. If this interval has length below *prec*, then the upper bound of this interval gives a numerical estimate of $\underline{\delta}$ with precision *prec*. If the length of the interval is larger than *prec*, then the algorithm continues refining the interval.

In the algorithm the range for $\underline{\delta}$ and the partition of this range is refined inductively. Given a range and partition we always start checking whether $\underline{\delta}$ belongs to the smallest interval. The lower and upper bound of this smallest interval will be denoted by h_l and h_u.

6.1 General method

We start with the initial partition $\mathcal{P} = \{(-\infty, 0], (0, \delta_{K-1}], \ldots, (\delta_{L+1}, \delta_L]\}$ and verify if the lower confidence bound is in $(-\infty, 0]$, the smallest interval of the partition $\mathcal{P}$. To this end we compute $p^{(2)}(0)$, $A(0)$ and $B(0)$. We distinguish three cases.

(A0) If $p^{(2)}(0) \leq B(0)$, then $p^{(2)}(h) \leq B(h) \leq \epsilon(h)$ for all $h \leq 0$, because $p^{(2)}(h)$ is decreasing and $B(h)$ is increasing for decreasing h. This implies that $\underline{\delta} \geq 0$ and hence we can reject the whole interval $(-\infty, 0]$. Therefore we remove this interval from the partition $\mathcal{P}$ and proceed with the next interval $(0, \delta_{k-1}]$, i.e. we put $h_l = 0$ and $h_u = \delta_{k-1}$

(B0) If $p^{(2)}(0) > A(0) + B(0) = \epsilon(0)$, then we must accept H_0 and hence $\underline{\delta} < 0$. We must therefore restrict attention to the interval $(-\infty, 0]$ and must not search in any other interval of the partition. In order to proceed we split the interval $(-\infty, 0]$ into the two subintervals $(-\infty, -\widetilde{\delta}]$ and $(-\widetilde{\delta}, 0]$, where $\widetilde{\delta}$ is some prefixed constant. In the program we use

$$\widetilde{\delta} = \sum_{j=1}^{k-1} \frac{\delta_j}{(k-1)} \tag{12}$$

the mean of the α-absorbing constants. We continue with the new partition $\mathcal{P} = \{(-\infty, -\widetilde{\delta}], (-\widetilde{\delta}, 0]\}$ and the lower interval $(-\infty, -\widetilde{\delta}]$, i.e., we set $h_l = -\infty$ and $h_u = -\widetilde{\delta}$.

(C0) If $B(0) < p^{(2)}(0) \leq A(0) + B(0)$ then we cannot decide whether $\underline{\delta}$ is in $(-\infty, 0]$ or not. In order to decide, we split the interval $(-\infty, 0]$ into the two subintervals $(-\infty, -\widetilde{\delta}]$ and $(-\widetilde{\delta}, 0]$, where $\widetilde{\delta}$ is defined in (12). As in case (B0) we continue with the interval $(-\infty, -\widetilde{\delta}]$, i.e., we put $h_l = -\infty$ and $h_u = -\widetilde{\delta}$. We keep, however, the other intervals of the initial partition, i.e., the new partition is

$$\mathcal{P} = \{(-\infty, -\widetilde{\delta}], (-\widetilde{\delta}, 0], (0, \delta_{K-1}], \ldots, (\delta_{L+1}, \delta_L]\}.$$

Whatever case (A0), (B0) or (C0) applies at the first step, we do the following with the current partition $\mathcal{P}$ and its lowest interval $(h_l, h_u]$.We compute $p^{(2)}(h_u)$, $A(h_l)$, $A(h_u)$ and $B(h_u)$. Again we have to distinguish between three cases.

(A) If $p^{(2)}(h_u) \leq B(h_u) + A(h_l)$ then $p^{(2)}(h) \leq B(h_u) + A(h_l) \leq \epsilon(h)$ for all h with $h_l < h \leq h_u$. This implies that $\underline{\delta} \geq h_u$ and hence we can reject the whole interval $(h_l, h_u]$. We therefore remove this interval from $\mathcal{P}$ and continue with the next higher interval and set h_l and h_u to the limits of the lowest interval from the new partition $\mathcal{P}$.

(B) If $p^{(2)}(h_u) > B(h_u) + A(h_u) = \epsilon(h_u)$, then we have to accept the interval. Therefore $\underline{\delta}$ must be in the interval $(h_l, h_u]$. If $h_u - h_l < prec$ then we stop the algorithm and report the interval $\underline{\delta} = h_u$. If $h_u - h_l \geq prec$ then we split the interval into $(h_l, \widetilde{h}]$, $(\widetilde{h}, h_u]$ where

$$\widetilde{h} = \begin{cases} h_u - \widetilde{\delta} & \text{if} \quad h_l = -\infty \\ \frac{h_u - h_l}{2} & \text{if} \quad h_l > -\infty \end{cases} \tag{13}$$

with $\widetilde{\delta}$ defined in (12). We continue with the partition $\mathcal{P} = \{(h_l, \widetilde{h}], (\widetilde{h}, h_u]\}$ and its lowest interval $(h_l, \widetilde{h}]$, i.e., we leave h_l unchanged and set $h_u = \widetilde{h}$.

(C) If $B(h_u) + A(h_l) < p^{(2)}(h_u) \leq B(h_u) + A(h_u)$ then we cannot decide whether $\underline{\delta}$ is in the interval $(h_l, h_u]$ or not. Hence, we split $(h_l, h_u]$ into the two subintervals $(h_l, \widetilde{h}], (\widetilde{h}, h_u]$ with $\widetilde{h}$ as in (13). We replace $(h_l, h_u]$ by the two subintervals in the partition $\mathcal{P}$ and keep all other intervals unmodified.

We continue with case (A) to (C) until we end up in case (B) and the difference $h_u - h_l$ is smaller or equal to the pre-specified precision *prec*.

Remark

The reason why the algorithm always converges is that $B(h) \to 1$ and $p^{(2)}(h) \to 0$ for $h \to -\infty$. From this we can follow that at some point we must have $p^{(2)}(h_u) < B(h_u)$ which allows us to conclude case (A). From this point on we have a finite lower limit for $\underline{\delta}$. The convergence of the algorithm then follows from the continuity of $A(h)$ and $B(h)$ on the intervals $(\delta_k, \delta_{k-1}]$.

6.2 More efficient algorithm

Note that for the calculation of $B(h)$ we first need to determine $b_{k(h)}(h)$ which requires numerical integration and root finding. Hence, the computation of $B(h_u)$ at every step of the algorithm slows down the algorithm considerably. The numerical precision of $B(h)$ required for the steps (A) to (C) depends on how close $p^{(2)}(h)$ is to $\epsilon(h)$. If $p^{(2)}(h)$ is far off from $\epsilon(h)$ then $p^{(2)}(h) > \epsilon(h)$ can be decided also with a rough estimate for $B(h)$. In order to exploite this fact we do not always compute $b_{k(h)}(h)$ with maximum precision, but use a lower and upper bound $b_1(h) \leq b_{k(h)}(h) \leq b_2(h)$ of $b_{k(h)}(h)$ which is refined only if necessary. The bounds $b_1(h)$ and $b_2(h)$ imply lower and upper bounds $B_1(h) \geq B(h) \geq B_2(h)$ for $B(h)$ where

$$B_i(h) = P_h\left(\bigcap_{j=L+1}^{k(h)-1}\{Z_j < b_j\} \cap \left\{Z_{k(h)} \geq b_i(h)\right\} | Z_1 = z_1, \dots, Z_L = z_L\right).$$

We always start with the initial lower and upper bounds

$$\begin{aligned} b_1(h_u) &= \Phi^{-1}(1-\alpha) + h_u\sqrt{I_K}, \\ b_2(h_u) &= \Phi^{-1}(1-\alpha+A(h_u)) + h_u\sqrt{I_K} \end{aligned} \tag{14}$$

Then we replace $B(h_u)$ in the steps (A0) and (A) by $B_2(h_u)$, and in steps (B0) and (B) by $B_1(h_u)$. In the steps (C0) and (C) we replace $B(h_u)$ on the right side of the inequality by $B_1(h_u)$ and on the left side by $B_2(h_u)$. Note that all replacements lead to sharper inequalities, so that the new cases do no longer cover the whole sample space. Should we not be able to decide between these three modified cases then we refine the upper and lower bound $b_1(h_u)$ and $b_2(h_u)$ in the following way: we compute

$$b^*(h_u) = \tfrac{b_1(h_u)+b_2(h_u)}{2}$$

and the rejection probability (10) with $b_{k(h)(h)}$ replaced by $b^*(h_u)$. If this probability is above α then we replace $b_1(h_u)$ by $b^*(h_u)$, otherwise we replace $b_2(h_u)$ by $b^*(h_u)$. The other $b_i(h_u)$ remains unchanged.

To give the algorithm in detail, suppose that at step k of the algorithm we have the lower and upper bound $b_1(h_u)$ and $b_2(h_u)$ and the partition $\mathcal{P}$ with the lowest interval $(h_l, h_u]$. In order to decide whether (A), (B) or (C) is true we execute the following two steps consecutively.

(S1) We distinguish between three cases:

- (a) If $p^{(2)}(h_u) \leq A(h_l) + B_2(h_u)$ then we can conclude (A) and hence can reject the whole interval $(h_l, h_u]$. We remove this interval from the initial partition $\mathcal{P}$ and continue with the lowest interval and redefine correspondingly h_l and h_u.
- (b) If $p^{(2)}(h_u) > A(h_l) + B_1(h_u)$ then we can exclude case (A) and pass directly to the second step (S2) below, to verify (B) or (C).

   (c) If $A(h_l) + B_2(h_u) < p^{(2)}(h_u) \leq A(h_l) + B_1(h_u)$ then we cannot make any conclusion yet, because $b_1(h_u)$ and $b_2(h_u)$ are too rough estimates. Before improving them, we pass to step (S2), now without excluding (A).

(S2) In step (S2) we distinguish between the following three cases:

   (a) If $p^{(2)}(h_u) > A(h_u) + B_1(h_u)$ then we can conclude (B). This implies that $\underline{\delta}$ must be in the interval $(h_l, h_u]$. If $h_l - h_u < prec$ then we stop and put $\underline{\delta} = h_u$. Otherwise we split the interval $(h_l, h_u]$ as in (B) and start again with step (S1) for the partition $\mathcal{P} = \{(h_l, \widetilde{h}], (\widetilde{h}, h_u]\}$ and $b_1(\widetilde{h})$, $b_2(\widetilde{h})$ as in (14) with h_u replaced by $\widetilde{h}$.

   (b) If $A(h_l) + B_1(h_u) < p^{(2)}(h_u) \leq A(h_u) + B_2(h_u)$ then we can conclude (C) and refine the partition $\mathcal{P}$ as in (C) and continue to (S1) with this partition and $b_1(\widetilde{h})$ and $b_2(\widetilde{h})$ as in (14).

   (c) If neither (a) nor (b) is satisfied then we must improve the boundaries $b_1(h_u)$ and $b_2(h_u)$. We do this as explained before. If (A) was excluded at step (S1), then we execute step (S2) with the improved boundaries $b_1(h_u)$ and $b_2(h_u)$. If (A) was not excluded we continue with step (S1).

We repeat the whole procedure until we end up with case (B) and $h_u - h_l < prec$.

7 Integration of the C-source code into R

The presented algorithm is implemented in C, but designed for the integration in the R project for statistical computing. In the following the integretion of our C-source is described.

First create the C-files (asoCbound.c, functions.c, functions.h) given in Appendix A. Afterwards copy the files to your local folder on your Linux-system. Now the C-source has to be compiled for R by typing in the command prompt

Listing 1: Build shared library for dynamic loading

```
R CMD SHLIB [options] [-o libname] files
```

In our example the command is

```
R CMD SHLIB -L/usr/local/lib asoCbound.c functions.c -lgsl -lgslcblas
-lm
```

This command compiles the given source files and links all specified object files into a shared library named asoCbound.so. This shared library can be loaded into R using dyn.load or library.dynam. The R-function dyn.load() loads or unloads shared libraries and tests if a C-function is available. In our example the command is

dyn.load("asoCbound.so")

After the libraries are loaded into R, all the variables have to be passed to the compiled C-functions. This can be performed with the R-function

.C(name,..., NAOK = FALSE, DUP = TRUE, PACKAGE)

The first character string (name) gives the name of the C-function. In our case this would be the function main(); see the example below. Afterwards the variables which are passed to the C-function main() are given. It is important to notice that the types and names of the variables used in R must be the same as in C. Otherwise R will produce a fatal error. Different vector types in R can be modified for the C-function by the commands

as.integer()	integer vector
as.numeric()	real number vector
as.character()	string vector
as.logical()	logical vector

The function .C() will return a list of objects which contains the input variables as well as the output of the C-function. All variables are saved in the *out* object. Now the result can be shown by the command *out\$erg*.

It is convenient to create an R-file which automatically calls the C-files and passes the variables. In our example the R-file would be:

Listing 2: Example R-file

```
asoCbound<-function(pT,sT,iD,algl){
dyn.load("asoCbound.so")

e<-c(0,0)

out <- .C("main",
             k = as.integer(length(pT$a)),
             a = as.numeric(pT$a),
             b = as.numeric(pT$b),
             t = as.numeric(pT$t),
             theta = as.numeric(algl),
             level = as.numeric(pT$al),
             L = as.integer(iD$L),
             z = as.numeric(iD$z),
             k2 = as.integer(length(sT$a2)),
             a2 = as.numeric(sT$a2),
             b2 = as.numeric(sT$b2),
             t2 = as.numeric(sT$t2),
             T2 = as.integer(sT$T2),
             zT = as.numeric(sT$z2),
             erg = as.numeric(e))

out;

}
```

The function asoCbound() modifies the input variables and passes them directly to the C-function in the Appendix A. The call of this C-function would be

int main(int k, var a, var b, var t, var theta, double level, int L, double z, int k2, var a2, var b2, var t2, int T2, double zT, double erg).

7.1 Data preparation and call of the function

We continue with the example from section 2.4.1 and show how to prepare the data and call the implemented function. First we create an object of the given primary trial. This object consists of a varible al which specifies the level and the variables a_j and b_j for $j = 1, \ldots, K$ which specify the lower and upper stopping boundaries. These boundaries can be calculated as described in section 2.3. Furthermore we have to declare the information times t_j which can be calculated by

$$t_j = \tfrac{n_j}{4\sigma^2} \qquad \text{for} \qquad j = 1, \ldots, K$$

where n_j was the number of patients recruited between stage $j-1$ and stage j. Now the objects for the primary trial is

```
//Object of the primary trial
pT=list(al=0.025,
        t=0.06125*c(1:4),
        a=rep(-8,4),
        b=c(3.155,2.818,2.439,2.014))
```

On the bases of the primary trial we can calculate the α-absobing constants as in (9)

```
//Alpha-absorbing parameters
algl=c(4.830182,2.331478,0.986098,0.000000)
```

Now we perform a design adaptation at look $L = 1$. This data is stored in the interim data object iD. It comprises of the stage of the adaptation L and the z-statistic observed at this stage

```
//Object of the interim data
iD=list(L=1,z=0.742)
```

After this design adaptation we first have to calculate the conditional error $\epsilon(0)$ as described in section 2.2. Now we can create an object for the secondary trial sT at level $\epsilon(0)$. The secondary trial consists of the lower and upper stopping boundaries $a_j^{(2)}$ and $b_j^{(2)}$ for $j = 1, \ldots, K^{(2)}$ and information times

$$t_j^{(2)} = \frac{n_j^{(2)}}{n_1} t_1$$

where $n_j^{(2)}$ is the number of patients recruited between stage $j-1$ and j of the secondary trial. As given in the example we assume that the secondary trial stops at stage $T^{(2)}$ with the observed test-statistic $z_{T^{(2)}}^{(2)}$. This data are also stored in the sT-object.

```
//Object of the secondary trial
sT=list(t2= 0.158125*c(1:4),
        a2=rep(-8,4),
        b2=c(3.093995,2.749179,2.360214,1.921324),
        T2=3,
        z2=2.755)
```

Now we can pass the objects to the function *asoCbound*(),

```
asoCbound(pT,sT,iD,algl)
```

which calculates for the given primary trial, α-absorbing parameters, interim data and secondary trial the corresponding stage wise adjusted confidence interval.

8 Simulation study

We used the C-program given in the Appendix A for a simulation study for the investigation of the coverage probability of the stage-wise confidence bound and median of the corresponding estimate. Furthermore, we want to compare the stage-wise adjusted confidence intervals and the repeated confidence intervals with regard to their coverage probability. We calculate the repeated confidence intervals only at the stage $T^{(2)}$ where the trial stops.

We start with a three-look, one-sided group sequential design at level 0.025 with equally spaced looks and without stopping for futility. We test $H_0 : \delta \leq 0$ and assume a known standard deviation $\sigma = 1$. The initial maximum sample size $N_{\max} = 390$ was chosen to detect a mean difference of $\delta = 0.3$ with 90% power. We used the O'Brien-Fleming (1979) boundaries for the initial design. We perform design adaptations at the first look of the primary trial, $L = 1$.

For the secondary trial we assume a maximal sample size of $N^{(2)}_{\max} = 520$ and a minimal sample size of $N^{(2)}_{\min} = 260$. For sample size reassessment we re-estimate the effect by the average $\hat{\theta}$ of the original δ and the interim estimate $\hat{\delta}_1$ of δ, i.e., $\hat{\theta} = \frac{\delta+\hat{\delta}_1}{2}$. Based on this estimate we define two rules:

1. If $\hat{\theta} \leq 0$ then we keep the initial design.
2. If $\hat{\theta} > 0$ we adapt the sample size and calculate the total sample size of the secondary trial by

$$N^{(2)} = \min(N^{(2)}_{\max}, \max(N^{(2)}_{\min}, \tfrac{4\sigma^2(\Phi^{-1}(\Pi)-\Phi^{-1}(\epsilon))^2}{\hat{\theta}^2}))$$

   where Π is the conditional power and ϵ the conditional rejection probability.

For simplicity the sample size is computed with the usual fixed sample size formula ignoring the group sequential nature of the secondary trial. The number of stages of the secondary trial was chosen such that the number of patients recruited between each look, $\frac{N^{(2)}}{K^{(2)}}$, is just below 130. We also used O'Brien-Fleming boundaries for the secondary trial. Each trial was simulated 10.000 times for several different true δ values.

The results of the simulations are given in Table 1 and Table 2. In Table 1 the coverage probabilities of the 97.5% stage-wise adjusted confidence intervals and the 97.5% repeated confidence intervals are given. Table 2 shows the median, mean and standard error of the 10.000 point estimates for the stage-wise adjusted and repeated method. It can easily be seen that the stage-wise adjusted confidence intervals provide for all values of δ exact coverage probability up to the simulation error. However the repeated confidence intervals have only for $\delta = 0$ exact coverage and become increasingly conservative for higher values of δ. Furthermore, the stage-wise adjusted confidence interval method produces point estimates that are median unbiased for all values of δ, whereas the repeated confidence interval method produces point estimates that are only median unbiased for $\delta = 0$, becoming increasingly negatively biased for higher values of δ. The standard error of the point estimates are similar.

Confidence Intervals			
Spending function	True δ	Actual Coverage of 97.5% Confidence Intervals	
		SWACI	RCI
$\gamma(-4)$	-0.2	0.9737	0.9756
$\gamma(-4)$	0.0	0.9762	0.975
$\gamma(-4)$	0.1	0.9769	0.9924
$\gamma(-4)$	0.2	0.9764	0.9947
$\gamma(-4)$	0.3	0.9638	0.9968
$\gamma(-4)$	0.4	0.9729	0.9984
$\gamma(-4)$	0.5	0.9729	0.9998

Table 1: Confidence intervals; 10.000 simulations; 3-look primary trial with adaptation at look 1; 2- to 4-look secondary trial

Point Estimates							
Spending function	True δ	Median of $\underline{\delta}_{0.5}$		Mean of $\underline{\delta}_{0.5}$		Standard Error of $\underline{\delta}_{0.5}$	
		SWACI	RCI	SWACI	RCI	SWACI	RCI
$\gamma(-4)$	-0.2	-0.2010	-0.2319	-0.2010	-0.2316	0.001	0.001
$\gamma(-4)$	0.0	-0.0005	-0.0350	-0.0046	-0.0385	0.0009	0.0009
$\gamma(-4)$	0.1	0.0996	0.0666	0.1029	0.0702	0.0009	0.0009
$\gamma(-4)$	0.2	0.1986	0.1745	0.2130	0.1805	0.0010	0.0009
$\gamma(-4)$	0.3	0.3088	0.2695	0.3165	0.2796	0.0011	0.0011
$\gamma(-4)$	0.4	0.3962	0.3541	0.4156	0.3727	0.0013	0.0011
$\gamma(-4)$	0.5	0.4983	0.4562	0.5188	0.4661	0.0014	0.0012

Table 2: Point Estimates; 10.000 simulations; 3-look primary trial with adaptation at look 1; 2- to 4-look secondary trial

A Appendix

Listing 3: functions.h

```
typedef double *var;

//Struct of the primary trial object
typedef struct{
    double level;         //level
    double *a;            //lower bound
    double *b;            //upper bound
    double *t;            //information
    int k;                //number of stages
    double *theta;        //alpha_globbings
    double bonf_algl;     //bonferroni estimate
    }pT_obj;

//Struct of the secondary trial object
typedef struct{
    double *a;            //lower bound
    double *b;            //upper bound
    double *t;            //information
    int k;                //number of stages
    double zT;            //z-statistic at stage T
    int T;                //stage where the trial stops
    }sT_obj;

//Struct of the interim data object
typedef struct{
    int L;                //stage of the adaption
    double z;             //z-statistic at stage L
    }iD_obj;

//Struct of the H object
//describtion: contains all variables for the calculation of
//             the algorithm
typedef struct{
        double hl; //lower interval bound
        double hu; //upper interval bound
        double bl; //lower estimation of the threshold boundary (8)
        double b2; //upper estimation of the threshold boundary (8)
        double Al; //value of A at the lower interval bound
        double Au; //value of A at the upper interval bound
        double A;  //conditional rejection probability
        double B1; //upper bound for Bk
        double B2; //lower bound for Bk
        double p2; //p-value of secondary trial
        int test;  //result of the test
        }H_list;


//definitions of the different functions

H_list initial_H(H_list H);

iD_obj initial(iD_obj iD);

var alloc_var(int sizei);

pT_obj alloc_seqmon_obj(pT_obj pT,int size_i);

double seqmon(double *a,double *b,double *t,int k,double theta);

double seqmon_b(double *a,double *b,double *t,int k,double theta);

double alpha_glob (double x, void *params);

int  alpha_globbing(pT_obj pT);

int  theta_interval(pT_obj pT,double h);

double cerror(double theta_c,pT_obj pT,iD_obj iD,int calc_b);

double epsilon(double theta_a,pT_obj pT,iD_obj iD,double x);

double B(double theta_a,pT_obj pT,iD_obj iD,double x);

double A(double theta_a,pT_obj pT,iD_obj iD);

sT_obj alloc_sT_obj(sT_obj s,int size_i);
```

```
double p2(sT_obj sT,double theta);

double sword(double theta,int k,pT_obj pT,double x);
```

Listing 4: functions.c

```
#include <R.h>             //specific R library
#include <Rdefines.h>      //specific R library

#include <stdlib.h>
#include <stdio.h>
#include <math.h>
#include <gsl/gsl_roots.h>            //GSL
#include <gsl/gsl_errno.h>            //GSL
#include <gsl/gsl_math.h>             //GSL
#include "gsl/gsl_cdf.h"              //GSL

#include "functions.h"                //header-file


//all variables of the H_list object are initialised to 0.
//For detail of the H_list object see file functions.h
H_list initial_H(H_list H){
        H.hl=0;
        H.hu=0;
        H.b1=0;
        H.b2=0;
        H.Al=0;
        H.Au=0;
        H.A=0;
        H.B1=0;
        H.B2=0;
        H.p2=0;
        H.test=0;
        return H;
        }

//the iD_obj object is initialised to 0. For detail of
//the iD_obj object see file functions.h
iD_obj initial(iD_obj iD){
        iD.L=0;
        iD.z=0;
        };
```

```
//memory is allocated for the variables of the secondary trial
sT_obj alloc_sT_obj(sT_obj s,int size_i){
    int i=0;
    s.a=alloc_var(size_i);
    s.b=alloc_var(size_i);
    s.t=alloc_var(size_i);
    //init to 0
    for(i=0;i<size_i;i++){
            s.a[i]=0;
            s.b[i]=0;
            s.t[i]=0;
            }
    return s;
    }


//**************************    p2    ***************************
//description:    Calculates the p-value of the secondary trial
//
//input:          sT          Object for the secondary trial
//                theta       Value of theta
//
//output:         erg         p-value of the secondary trial

double p2(sT_obj sT,double theta){
    int i=0;
    double erg=0;
    var b_new=alloc_var(sT.k);

    for(i=0;i<sT.k;i++)b_new[i]=sT.b[i];

    int new_T=sT.T;
    new_T--;
    if(sT.zT<b_new[new_T] && sT.T<sT.k){printf("Error: zT < b[T]");
                                    return 0;}
    else {
         //the secondary trial stops at the first stage
         if(sT.T==1){
              free(b_new);
              return 1-gsl_cdf_ugaussian_P(sT.zT-theta*sqrt(sT.t[0]));
```

```
                }
            else{
                b_new[new_T]=sT.zT;
                erg=seqmon(sT.a,b_new,sT.t,sT.T,theta);
                }
            }
        free(b_new);
        return erg;
        };

//**********************    theta_interval    ********************
//description:     Calculates which interval contains theta
//
//input:           pT        Object of the primary trial
//                 h         Value of theta
//
//output:          i         number of the interval that contains
//                           theta

int  theta_interval(pT_obj pT,double h){
  int i;
  for(i=0;i<pT.k;i++){
          if(h>pT.theta[i]+0.0000001){i++;return i;}
          }
  return i;
};


//***************************    B    *************************
//description:     Calculates the value of B(h)
//
//input:           theta_a       Value of theta
//                 pt            Object for the primary trial
//                 iD            Object for the interim data
//                 x             Value of the last critical boundary
//
//output:          B             value of B(h)

double B(double theta_a,pT_obj pT,iD_obj iD,double x){
int i=0;
double B=0;
```

```
int interval=theta_interval(pT,theta_a);

int last=interval;
last--;
pT_obj pT_new;

pT_new.k=interval;

pT_new=alloc_seqmon_obj(pT_new,interval);
  if (!pT_new.a || !pT_new.b|| !pT_new.t|| !pT_new.theta )
    {
      printf("No memory free for double-components (B)\n");
      exit(1);
    }

for(i=0;i<interval;i++){
        pT_new.b[i]=pT.b[i];
        pT_new.a[i]=pT.a[i];
        pT_new.t[i]=pT.t[i];
        }

pT_new.b[last]=x;

B=cerror(theta_a,pT_new,iD,1);

//free memory
free(pT_new.a);free(pT_new.b);free(pT_new.t);free(pT_new.theta);

return B;
};

//**************************** A ***************************
//description:      Calculates the value of A(h)
//
//input:            theta_a       Value of theta
//                  pt            Object for the primary trial
//                  iD            Object for the interim data
//
//output:           A             value of A(h)

double A(double theta_a,pT_obj pT,iD_obj iD){
```

```
int i=0;
double A=0;
int interval=theta_interval(pT,theta_a);

if(interval-iD.L<=1)return 0;

else{

interval--;

pT_obj pT_new;
pT_new.k=interval;

pT_new=alloc_seqmon_obj(pT_new,interval);
  if (!pT_new.a || !pT_new.b|| !pT_new.t|| !pT_new.theta )
    {
      printf("No memory free for double-components (A)\n");
      exit(1);
    }

for(i=0;i<interval;i++){
        pT_new.b[i]=pT.b[i];
        pT_new.a[i]=pT.a[i];
        pT_new.t[i]=pT.t[i];

        }


A=cerror(theta_a,pT_new,iD,0);

//free memory
free(pT_new.a);free(pT_new.b);free(pT_new.t);free(pT_new.theta);

return A;
}
};

//************************    cerror    ************************
//description:    Calculates the conditional error probability
//
//input: theta_c     Value of theta
```

```
//         pT          Object for the primary Trial
//         iD          Object for the interim Data
//         calc_b      if 0   claculate seqmon
//                     if 1   claculate seqmon_b
//
//output: e            conditional error probability

double cerror(double theta_c,pT_obj pT,iD_obj iD,int calc_b){
int i=0;
double e=0;
int k_new=pT.k;
k_new-=iD.L;
int L=iD.L;
L--;

if(k_new<1){
            printf("cannot compute cerror for L>K");
            return 0;
            }

var a_new=alloc_var(k_new);
var b_new=alloc_var(k_new);
var t_new=alloc_var(k_new);

int c=0;
for(i=iD.L;i<pT.k;i++){
        b_new[c]=(pT.b[i]*sqrt(pT.t[i])-iD.z*sqrt(pT.t[L]))/
                  sqrt(pT.t[i]-pT.t[L]);
        a_new[c]=pT.a[i];
        t_new[c]=pT.t[i]-pT.t[L];
        c++;
        }

if(!calc_b)
e=seqmon(a_new,b_new,t_new,k_new,theta_c);
else
e=seqmon_b(a_new,b_new,t_new,k_new,theta_c);

//free memory
free(a_new);free(b_new);free(t_new);
```

```
return e;
};


//************************   alloc_var   *********************
//description:      Allocates unused space for an array
//
//input:            sizei     size of the allocated array
//
//output:           V         the new allocated array

var alloc_var(int sizei){
int i=0;
var V = (double *) calloc(sizei , sizeof(double));
//init to 0
for(i=0;i<sizei;i++){
          V[i]=0;
          }
return V;
};


//********************   alloc_seqmon_obj   ******************
//description:      Allocates unused space for all variables of a
//                  pT_obj
//
//input:            pT        Object for the primary Trial
//                  sizei     size of the allocated array
//
//output:           pT        A pT_obj with the allocated variables

pT_obj alloc_seqmon_obj(pT_obj pT,int size_i){
    int i=0;
    pT.a=alloc_var(size_i);
    pT.b=alloc_var(size_i);
    pT.t=alloc_var(size_i);
    pT.theta=alloc_var(size_i+1);
    //init to 0
    for(i=0;i<size_i;i++){
          pT.a[i]=0;
          pT.b[i]=0;
```

```
            pT.t[i]=0;
            pT.theta[i]=0;
            }
    pT.theta[size_i]=0;
    return pT;
    };


//*************************       seqmon    ************************
//description:        computes the probabilities of crossing
//                    boundaries in a group sequential clinical
//                    trial. It implements the Armitage-McPherson
//                    and Rowe (1969) algorithm using the method
//                    described in Schoenfeld D. (2001).
//
//input:              a          lower boundaries
//                    b          upper boundaries
//                    t          information
//                    k          number of stages
//                    theta      value of theta
//
//output:             pU_last    probability of crossing the
//                               boundaries

double seqmon(double *a,double *b,double *t,int k,double theta){

  int c=1;
  int interval=500;
  var M=alloc_var(interval);
  var Ma=alloc_var(interval);
  var Maf=alloc_var(interval);
  var x0=alloc_var(interval);
  var d=alloc_var(k);
  var pU=alloc_var(k);
  var pL=alloc_var(k);
  var VU=alloc_var(interval);
  var VL=alloc_var(interval);
  var x=alloc_var(interval);
  int i=0;
  int j=0;
  int w=0;
```

```
var b_new=alloc_var(k);

if (!x || !VU|| !VL|| !x0 || !Maf ||!Ma||!M ||!d||!pU||!pL||!b_new)
  {
    printf("No memory free for double-components (seqmon)\n");
    exit(1);
  }

double pUd,pLd,VUf,VLf,VUM=0,VLM=0;

      for(i=0;i<k;i++){
   if(theta) b_new[i]=b[i]-theta*sqrt(t[i]);
   else b_new[i]=b[i];
   d[i]=(b_new[i]-a[i])/(interval);
   }

c=1;

for(i=0;i<interval;i++){
    x0[i]=a[0]+(c-0.5)*(d[0]);
    M[i]=(d[0]/sqrt(2*M_PI))*
          exp(-(pow((sqrt(t[0])*x0[i]),2))/(2*t[0]));
    c++;
    }

pUd=-(sqrt(t[0])*b_new[0])/sqrt(t[0]);
//gsl_cdf_ugaussian_P cumulative distribution functions
//For details see GNU Scientific Library
pU[0]=gsl_cdf_ugaussian_P(pUd);


pLd=(sqrt(t[0])*a[0])/sqrt(t[0]);
pL[0]=gsl_cdf_ugaussian_P(pLd);

int count=0;
int last=k;
last--;
for(i=1;i<k;i++){
  c=1;
   for(j=0;j<interval;j++){
```

```
            VUf=-(sqrt(t[i])*b_new[i] + (-sqrt(t[count])*x0[j]))/
                  sqrt(t[i]-t[count]);
            VU[j]=gsl_cdf_ugaussian_P(VUf);
            VLf=(sqrt(t[i])*a[i] + (-sqrt(t[count])*x0[j]))/
                 sqrt(t[i]-t[count]);
            VL[j]=gsl_cdf_ugaussian_P(VLf);
            VLM+=VL[j]*M[j];
            VUM+=VU[j]*M[j];
            x[j]=a[i] + (c - 0.5 ) * d[i];
            c++;
            }


        pL[i]=pL[count]+VLM;
        pU[i]=pU[count]+VUM;

        VLM=0.0;
        VUM=0.0;
      if(i!=last){
        for(w=0;w<interval;w++){
            for(j=0;j<interval;j++){
               Ma[j]=(d[i]*sqrt(t[i])/(sqrt(2.0*M_PI)*
                      sqrt(t[i] - t[count])))*
                      expl(-pow((sqrt(t[i])*x[w]-sqrt(t[count]) *
                      x0[j]),2)/(2 * (t[i] - t[count])));
                  Maf[w]+=Ma[j]*M[j];
               }
                  }
        for(j=0;j<interval;j++){
            M[j]=Maf[j];
            x0[j]=x[j];
            Maf[j]=0.0;
            Ma[j]=0.0;
            }
        }
        count++;
      }



   for(i=0;i<k;i++){
```

```
                if(!pU[i] || !pL[i]){
                        free(M);free(Ma);free(Maf);free(x0);
                        free(VU);free(VL);free(x);free(d);
                        free(pU);free(pL);free(b_new);
                        return -1;
                        }
                }

double pU_last=pU[last];

free(M);free(Ma);free(Maf);free(x0);free(VU);free(VL);free(x);
free(d);free(pU);free(pL);free(b_new);

return pU_last;

};


//*************************    seqmon_b    ***********************
//description:          It's a modified version of seqmon and
//                      calculates the probability of crossing
//                      only the last upper boundary
//
//input:                a           lower boundaries
//                      b           upper boundaries
//                      t           information
//                      k           number of stages
//                      theta       theta
//
//output:               pU_last     probability of crossing the last
//                                  boundary


double seqmon_b(double *a,double *b,double *t,int k,double theta){

  int c=1;
  int interval=500;
  var M=alloc_var(interval);
  var Ma=alloc_var(interval);
  var Maf=alloc_var(interval);
  var x0=alloc_var(interval);
var d=alloc_var(k);
var VU=alloc_var(interval);
var x=alloc_var(interval);
var b_new=alloc_var(k);

double pU,pUd;
int i=0;
int j=0;
int w=0;

if (!x || !VU|| !x0 || !Maf ||!Ma||!M ||!d||!b_new)
  {
    printf("No memory free for double-components (seqmon_b)\n");
    exit(1);
  }

double VUf=0,VUM=0;

      for(i=0;i<k;i++){
   if(theta) b_new[i]=b[i]-theta*sqrt(t[i]);
   else b_new[i]=b[i];
   d[i]=(b_new[i]-a[i])/(interval);
   }

c=1;



for(i=0;i<interval;i++){
    x0[i]=a[0]+(c-0.5)*(d[0]);
    M[i]=(d[0]/sqrt(2*M_PI))*exp(-(pow((sqrt(t[0])*x0[i]),2))/
          (2*t[0]));
    c++;
    }

pUd=-(sqrt(t[0])*b_new[0])/sqrt(t[0]);
pU=gsl_cdf_ugaussian_P(pUd);

int count=0;
int last=k;
```

```
if(k==1){
          free(M);free(Ma);free(Maf);free(x0);free(VU);free(x);
          free(d);free(b_new);
          return pU;
          }

last--;
for(i=1;i<k;i++){
   c=1;
   for(j=0;j<interval;j++){
        x[j]=a[i] + (c - 0.5 ) * d[i];
        c++;
        }
  if(i!=last){
    for(w=0;w<interval;w++){
          for(j=0;j<interval;j++){
             Ma[j]=(d[i]*sqrt(t[i])/
                   (sqrt(2.0*M_PI)*sqrt(t[i] - t[count])))*
                    exp(-pow((sqrt(t[i]) * x[w] - sqrt(t[count]) *
                    x0[j]),2)/(2 * (t[i] - t[count])));
                 Maf[w]+=Ma[j]*M[j];
             }
                 }
    for(j=0;j<interval;j++){
          M[j]=Maf[j];
          x0[j]=x[j];
          Maf[j]=0.0;
          Ma[j]=0.0;
          }
    }
  if(i==last){
    for(j=0;j<interval;j++){
          VUf=-(sqrt(t[i])*b_new[i] + (-sqrt(t[count])*x0[j]))/
                sqrt(t[i]-t[count]);
          VU[j]=gsl_cdf_ugaussian_P(VUf);
          VUM+=VU[j]*M[j];
          }
    pU=VUM;
    }
    count++;
 }
```

```
VUf=VUM=0;
free(M);free(Ma);free(Maf);free(x0);free(VU);free(x);free(d);
free(b_new);

if(!pU)return -1;
else return pU;

};




//***********************    sword    *************************
//description:    Calculates condtitonal rejection probability
//                with last critical boundary equal x
//
//input:          theta        value of theta
//                k            number of stages
//                pt           object for the primary trial
//                x            value of the last critical boundary
//
//output:         erg


double sword(double theta,int k,pT_obj pT,double x){
       int i=0;
       int new_k=k;
       double erg=0;
       new_k--;
         if(k==1){
              return 1-gsl_cdf_ugaussian_P(x-
                               theta*sqrt(pT.t[new_k]));
              }
         else {
              var b_new=alloc_var(k);
              var t_new=alloc_var(k);

              for(i=0;i<k;i++){
                     b_new[i]=pT.b[i];
                     t_new[i]=pT.t[i];
```

```
                        }
                b_new[new_k]=x;

                erg=seqmon(pT.a,b_new,t_new,k,theta);
                free(b_new);
                free(t_new);
                return erg;
                 }
};
```

Listing 5: asoCbound.c

```c
#include <R.h>              //specific R library
#include <Rdefines.h>       //specific R library

#include <stdio.h>
#include <gsl/gsl_math.h>   //GSL
#include <gsl/gsl_errno.h>  //GSL
#include "gsl/gsl_cdf.h"    //GSL

#include "functions.h" //header-file

#define INF -100        //pre-defined value for minus infinity
#define pprec 0.0001    //pre-defined value for the precision

/****************************************************************
//The function testint calculates if the tested intervals is
//rejected, accepted or that there is no decision possible.
//input: H      obejct of H_list
//       pT     object of the primary trial
//       sT     object of the secondary trial
//       iD     object of the interim data
//output: H
//        the output of the test is saved in
//        H.test(2=reject; 1=accept; 0=no decision)
/****************************************************************

H_list testint(H_list H,pT_obj pT,iD_obj iD,sT_obj sT){
    int intervalhl=0;
    int intervalhu=0;
    int interval=0;
    int last=0;
    int check_a=0;
    int check_b=0;
    int i=0;

    //theta_interval calculates the stage containing hl and hu
    intervalhl=theta_interval(pT,H.hl);
    intervalhu=theta_interval(pT,H.hu);
    //if hl and hu are not in the same interval we have to abort
    if(intervalhl!=intervalhu && H.hl>0 && intervalhu>iD.L){
    printf("hl and hu not in the same alpha-globbing interval or
            interval < iD L");
    H.test=-1;
    return H;
    }

    last=intervalhu;
    last--;

    //Value of A at the upper bound hu
    if(H.Au==0){H.Au=A(H.hu,pT,iD);}

    //P-value of the secondary trial
    if(H.p2==0)H.p2=p2(sT,H.hu);

//*************************************************************
//parameter values so small that we can neglect rejection
//boundaries from the first K-1 stages
    if(H.hu<=pT.bonf_algl){
        //gsl_cdf_ugaussian_Pinv inverse cumulative distribution
        //functions.
        //For details see GNU Scientific Library
        H.b1=H.b2=gsl_cdf_ugaussian_Pinv(1-pT.level)+
                H.hu*sqrt(pT.t[last]);
        if(!H.B1)H.B1=B(H.hu,pT,iD,H.b1);
      if(H.p2<=H.B1)H.test=2;
      else{
            if(H.p2>H.B1+H.Au)H.test=1;
            else H.test=0;
            }
      return H;
      }

//*************************************************************
//parameter values not small enough that we can neglect rejection
//boundaries
    else {
         //Value of A at the lower bound hu
         if(!H.Al){
```

```
                  if(H.hl==INF)H.Al=0; //if hl=-inf
                  else H.Al=A(H.hl,pT,iD);
                  }

             interval=last;
             interval--;

//***********************************************************
//hu is equal to the upper boundary of the tested interval
       if(fabs(pT.theta[interval]-H.hu)<0.00001 && H.hu>0.00001)
         {
         if(H.p2<=H.Al){H.test=2; // reject interval
               }
         else {
              if(H.p2<=H.Au){H.test=1;//accept interval
                   }
              else {H.test=0;//no decision possible
                   }
              }
         return H;
         }
//***********************************************************

             var a_new=alloc_var(last);
             var b_new=alloc_var(last);
             var t_new=alloc_var(last);

             if(!H.A){
                      for(i=0;i<last;i++){
                            a_new[i]=pT.a[i];
                            b_new[i]=pT.b[i];
                            t_new[i]=pT.t[i];
                            }
                    H.A=seqmon(a_new,b_new,t_new,last,H.hu);
                }

             free(a_new);
             free(b_new);
             free(t_new);
```

```
//lower and upper estimations for the boundary bk(h) as in (8)
if(!H.b1)H.b1=gsl_cdf_ugaussian_Pinv(1-pT.level)+

if(!H.b2)H.b2=gsl_cdf_ugaussian_Pinv(1-((pT.level-H.A)/(1-H.A)))+
                  H.hu*sqrt(pT.t[last]);

//Values of B at the upper and lower bound b1 and b2
if(!H.B1)H.B1=B(H.hu,pT,iD,H.b1);
if(!H.B2)H.B2=B(H.hu,pT,iD,H.b2);


//Algorithm
for(i=0;i<100;i++){

//step (S1)
if(!check_a){
     //case (a)
     if(H.p2<=H.Al+H.B2){
                              H.test=2;//reject interval
                              check_a=1;
                              break;
                              }
     //case (b)
     if(H.p2>H.Al+H.B1){
                             check_a=1;//exclude case (a) and pass
                                         //directly to (S2)
                           }
     //case (c)
     if(H.Al+H.B2<H.p2 && H.p2<=H.Al+H.B1){
                                   check_a=0;//Without excluding case (a),
                                               //pass directly to (S2)
                                   }
     }
//step (S2)
if(!check_b){
            //case (a)
            if(H.p2>H.Au+H.B1){
                                   check_b=1;
                                   H.test=1;//accept the interval
                                   break;
                              }
              //case (b)
              if(H.Al+H.B1<H.p2 && H.p2<=H.Au+H.B2){//refine interval
                             H.test=0;
                             check_b=1;
                             break;
                             }
              else{
                     //calculate rejection probability
                     if(sword(H.hu,intervalhl,pT,(H.b1+H.b2)/2)>pT.level){
                          H.b1=(H.b1+H.b2)/2;
                          H.B1=B(H.hu,pT,iD,H.b1);
                          }
                     else{
                          H.b2=(H.b1+H.b2)/2;
                          H.B2=B(H.hu,pT,iD,H.b2);
                          }
              }
  }
  }
  if(!H.test && i>98){
             //maximum number of 100 iterations reached without
             //convergence
             printf("maximum number of 100 iterations reached without
                         convergence");
             H.test=0;
             return H;
             }
  else return H;
  }

  };


  //description: The function main calculates from a given group
  //             sequential plan with adaptive design the associated
  //             intervals, passes them to the function testint and
  //             returns the lower confidence bound
  //
  //input:     parameters for the primary trial:
  //             k            number of stages
```

```
//              a          lower boundaries
//              b          upper boundaries
//              t          information
//              theta      theta
//              level      alpha
//          parameters for the interim data
//              L          stage of the adaption
//              z          z-statistic at stage L
//          parameters for the secondary trial
//              k2         number of stages
//              a2         lower boundaries
//              b2         upper boundaries
//              t2         information
//              T2         stage where the trial stops
//              zT         z-statistic at stage T2
//
//output:       erg        lower bound of the calculated confidence
//                         interval

int main(int *k,var a,var b,var t,var theta,double *level,
int *L,double *z,int *k2,var a2,var b2,var t2,int *T2,
double *zT,double *erg){

int i=0;

pT_obj pT;//declare primary trial object pT
iD_obj iD;//declare interim data object iD
iD=initial(iD);
sT_obj sT;//declare secondary trial object sT


pT.k=k[0];//stages pT


pT=alloc_seqmon_obj(pT,pT.k);
  if (!pT.a || !pT.b|| !pT.t|| !pT.theta )
    {
      //printf("No memory available for double-components!\n");
      exit(1);
    }
```

```
for(i=0;i<pT.k;i++){
          pT.a[i]=a[i];//lower boundaries pT
          pT.b[i]=b[i];//upper boundaries pT
          pT.t[i]=t[i];//information pT
          pT.theta[i]=theta[i];//alpha_globbings
          }


sT.k=k2[0];//stages sT

sT=alloc_sT_obj(sT,sT.k);
   if (!sT.a || !sT.b|| !sT.t )
      {
         //printf("No memory available for double-components!\n");
         exit(1);
      }

for(i=0;i<sT.k;i++){
          sT.a[i]=a2[i];//lower boundaries pT
          sT.b[i]=b2[i];//upper boundaries pT
          sT.t[i]=t2[i];//information sT
          }


iD.L=L[0];//stage of the design adaptations

iD.z=z[0];//z-statistic at stage L

sT.zT=zT[0];//z-statistic at stage T2

sT.T=T2[0];//stage where the secondary trials stops

pT.level=level[0];//alpha


int j=0;
double min_bonf=0;
int count =0;
int start=pT.k;
start-=2;
```

```
int thetac=pT.k;
thetac--;
int interval;
int test_i=0;
int test=0;
int testit=0;
int old_interval=0;


//initial object Hl
H_list Hl[pT.k][100];
for(i=0;i<pT.k;i++){
            for(j=0;j<100;j++){
            Hl[i][j]=initial_H(Hl[i][j]);
            }
}


double malgl=0;//mean of the alpha-globbing constants (6)
for(i=0;i<pT.k;i++){
            malgl+=pT.theta[i];
            }
malgl=malgl/pT.k;



//*******************     Bonferroni estimate     ******************

pT.bonf_algl=100;

for(i=0;i<pT.k-1;i++){
            min_bonf=(pT.b[i]-gsl_cdf_ugaussian_Pinv(1-pprec/i))/
                        sqrt(pT.t[i]);
            if(min_bonf<=pT.bonf_algl)pT.bonf_algl=min_bonf;
            }

//************************************************************


for(i=start;i>=0;i--){
```

```
count=0;
interval=0;

//bounds for the first interval
if(i==start){
       Hl[i][0].hl=INF;
       Hl[i][0].hu=pT.theta[i];
       }
//bounds for the rest of the intervals
else{
       Hl[i][interval].hl=pT.theta[thetac];
       Hl[i][interval].hu=pT.theta[i];
       }

Hl[i][interval]=testint(Hl[i][interval],pT,iD,sT);
count++;

if(Hl[i][interval].test==2){//the whole interval can be rejected
  }
else {//the interval is accepted or there is no decision possible
  old_interval=interval;
  interval++; //increment list-counter
  Hl[i][interval].Al=Hl[i][old_interval].Al;
  Hl[i][interval].hl=Hl[i][old_interval].hl;

  //only in the case of the first interval (7)
  if(i==start)Hl[i][interval].hu=Hl[i][old_interval].hu-malgl;
  else Hl[i][interval].hu=(Hl[i][old_interval].hl+
                   Hl[i][old_interval].hu)/2;

  //new subinterval is tested
  Hl[i][interval]=testint(Hl[i][interval],pT,iD,sT);
  count++;


  //continues as long as we donnot have the pre-defined precision
  while(fabs(Hl[i][interval].hu-Hl[i][interval].hl)>=pprec &&
        testit!=1){

        if(Hl[i][interval].test==2){ //reject the interval
             old_interval=interval;
        old_interval--;

        //in the case of the first interval
        if(i==start){
            Hl[i][interval].hl=Hl[i][interval].hu;
            Hl[i][interval].hu=Hl[i][old_interval].hu;
            Hl[i][interval].p2=0;
            Hl[i][interval].Al=0;
            Hl[i][interval].Au=0;
            Hl[i][interval].B1=0;
            Hl[i][interval].B2=0;
            Hl[i][interval].b1=0;
            Hl[i][interval].b2=0;
            }
        //in all other cases we can save a couple of values
        //to save computation time
        else {
            Hl[i][interval].hl=Hl[i][interval].hu;
            Hl[i][interval].hu=Hl[i][old_interval].hu;
            Hl[i][interval].B1=Hl[i][old_interval].B1;
            Hl[i][interval].B2=Hl[i][old_interval].B2;
            Hl[i][interval].b1=Hl[i][old_interval].b1;
            Hl[i][interval].b2=Hl[i][old_interval].b2;
            Hl[i][interval].Au=Hl[i][old_interval].Au;
            Hl[i][interval].Al=0;
            Hl[i][interval].p2=0;
            }
        }
else{// akzept or no decision
       interval++;//increment list-counter
       old_interval++;

       //in the case of the first interval
       if(i==start && Hl[i][interval].hl==INF){
            Hl[i][interval].hu=Hl[i][old_interval].hu-malgl;
            Hl[i][interval].p2=0;
            }
       else{
       Hl[i][interval].hl=Hl[i][old_interval].hl;
       Hl[i][interval].hu=(Hl[i][old_interval].hl+
                              Hl[i][old_interval].hu)/2;
```

```
                Hl[i][interval].Al=Hl[i][old_interval].Al;
                Hl[i][interval].p2=0;
                }
            }

            //test new defined interval
            Hl[i][interval]=testint(Hl[i][interval],pT,iD,sT);
            count++;

            if(fabs(Hl[i][interval].hu-Hl[i][interval].hl)<=0.001 &&
               Hl[i][interval].hu!=Hl[i][interval].hl){
                       testit=1;
                       }

    }//end while

    if(fabs(Hl[i][interval].hu-Hl[i][interval].hl)<=0.001 &&
       testit==1){
                break;
                }


  }//end else
  if(count>98){
                //printf("\n100 iterations reached");
                }

  thetac--;
  }//end for

  //free memory
  free(pT.a);
  free(pT.b);
  free(pT.t);
  free(pT.theta);

  free(sT.a);
  free(sT.b);
  free(sT.t);

  //save results in erg
```

```
erg[0]=Hl[i][interval].hl;
erg[1]=Hl[i][interval].hu;


}//end main
```

References

[Armitage, 1957] Armitage, P. (1957). Restricted sequential procedures. *Biometrika 44, 9-56.*

[Armitage et al., 1969] Armitage, P., McPherson, C., and Rowe, B. (1969). Repeated significanca tests on accumulating data. *J. Roy. Statist. Soc. A. 132, 235-244.*

[Armitage and Schneiderman, 1958] Armitage, P. and Schneiderman, M. (1958). Statistical problems in a mass screening program. *Ann. New York Academy Sci. 76, 896-908.*

[Bartky, 1943] Bartky, W. (1943). Multiple sampling with constant probability. *Ann. Math. Statist. 14, 363-377.*

[Bauer, 1989] Bauer, P. (1989). Multistage testing with adaptive designs (with discussion). *Beometrie und Informatik in Medizin Und Biologie , 130-148.*

[Bauer and Köhne, 1994] Bauer, P. and Köhne, K. (1994). Evalution of experiments withadaptive interim analyses. *Biomertrics 50, 1029-1041.*

[Brannath et al., 2007] Brannath, W., Mehta, C. R., and Posch, M. (2007). Exact confidence bounds following adaptive group sequential tests. *submitted.*

[Brannath et al., 2002] Brannath, W., Posch, M., and Bauer, P. (2002). Recursive combination tests. *Journal of the American Statistical Association 97, 236-244.*

[Brannath, 2006] Brannath, W. e. a. (2006). Estimation in flexible two stage designs. *Statistics in Medicine 25, 3366-3381.*

[CPMP, 1998] CPMP (1998). Note for guidance on statistical principles for clinical trias. *ICH/363/96.*

[Cui et al., 1999] Cui, L., Hung, H., and Wang, S.-J. (1999). Modification of sample size in group sequential clinical trials. *Biometrica 55, 853-857.*

[DeMets and Ware, 1980] DeMets, D. and Ware, J. (1980). Group sequential methods for clinical trials with one-sided hypothesis. *Biometrika 67, 651-660.*

[DeMets and Ware, 1982] DeMets, D. and Ware, J. (1982). Asymmetric group sequential boundaries for monitoring cinical trials. *Biometrika 69, 661-663.*

[Denne, 2001] Denne, J. (2001). Sample size recalculation using conditional power. *Statistics in Medizine 20, 2645-2660.*

[Dodge and Roming, 1929] Dodge, H. and Roming, H. (1929). A method for sampling inspection. *Bell Syst. Tech. J. 8, 613-631.*

[Dunnett, 1961] Dunnett, C. (1961). The statistical theory of drug screening. *Quantitative Methods in Pharmacology, Amsterdam: North-Holland, 212-231.*

[Eales and Jennison, 1992] Eales, J. and Jennison, C. (1992). An improved method for deriving optimal one-sided group sequential tests. *Biometrika 79, 13-24.*

[EAST-5, 2007] EAST-5 (2007). Software for the design and interim monitoring of flexible clinical trials. *Cytel Software Corporation, Cambridge, MA.*

[Elfring and Schultz, 1973] Elfring, G. and Schultz, J. (1973). Group sequential designs for clinical trials. *Biometrics 29, 471-477.*

[Hartung and Knapp, 2003] Hartung, J. and Knapp, G. (2003). A new class of completely self-designing clinical trials. *Biometrical Journal 45, 3-19.*

[Hwang et al., 1990] Hwang, I., Shih, W., and DeCani, J. (1990). Group sequential designs using a family of type i error probability spending functions. *Statistics in Medizine 9, 1439-1445.*

[Jennison, 1987] Jennison, C. (1987). Efficient group sequential tests with unpredictable group sizes. *Biometrika 74, 155-165.*

[Jennison and Turnbull, 1984] Jennison, C. and Turnbull, B. (1984). Repeated confidence intervals for group sequential clinical trials. *Contr. Clin. Trials 5, 33-45.*

[Jennison and Turnbull, 1989] Jennison, C. and Turnbull, B. (1989). Interim analyses: the repeated confidence interval approach (with discussion). *J.R. Statist. Soc. B 51, 305-361.*

[Jennison and Turnbull, 2000] Jennison, C. and Turnbull, B. (2000). Group sequential methods with applications to clinical trials. *Chapman Hall, Boca Raton, London, New York, Washington, D.C.*

[Kim and DeMets, 1987] Kim, K. and DeMets, D. (1987). Confidende intervals following group sequential tests in clinical trials. *Biometrics 43, 857-864.*

[Lan and DeMets, 1983] Lan, K. and DeMets, D. (1983). Discrete sequential boundaries for clinical trials. *Biometrica 70, 659-663.*

[Lehmacher and Wassmer, 1999] Lehmacher, W. and Wassmer, G. (1999). Adaptive sample size calculation in group sequential trials. *Beometrics 57, 886-891.*

[Madsen and Fairbanks, 1983] Madsen, R. and Fairbanks, K. (1983). P values for multistage and sequential tests. *Technometrics 25, 285-293.*

[Mehta et al., 2007] Mehta, C., Bauer, P., Posch, M., and Brannath, W. (2007). Repeated confidence intervals for adaptive group sequential trials. *Statistics in Medicine 26, 5422-5433.*

[Müller and Schäfer, 2001] Müller, H.-H. and Schäfer, H. (2001). Adaptive group sequential design for clinical trials: Combining the advantages of adaptive and of classic group sequential approaches. *Biometrics 57, 886-891.*

[Müller and Schäfer, 2004] Müller, H.-H. and Schäfer, H. (2004). A general statistical principle for changing a design any time during the course of a trial. *Statistics in Medicine 23, 2497-2508.*

[O'Brien and Fleming, 1979] O'Brien, P. and Fleming, T. (1979). A multiple testing procedure for clinical trials. *Biometrica 35, 549-556.*

[Pocock, 1977] Pocock, S. (1977). Group sequential methods in the design and analysis of clinical trials. *Biometrica 64, 191-199.*

[Proschan and Hunsberger, 1995] Proschan, M. and Hunsberger, S. (1995). Designed extensions of studies based on conditional power. *Biomertrics 51, 1315-1324.*

[Roseberry and Gehan, 1964] Roseberry, T. and Gehan, E. (1964). Operating characteristic curves and accept-reject rules for two and three stage screening procedures. *Biometrics 20, 73-84.*

[Schneiderman, 1961] Schneiderman, M. (1961). Statistical problems in the screening search for anticancer drugs by the national cancer institute of the united states. *Quantitative Methods in Pharmacology, Amsterdam: North-Holland, 232-246.*

[Shen and Fisher, 1999] Shen, J. and Fisher, L. (1999). Statistical inference for self-desining clinical trials with a one-sided hypothesis. *Biometrics 55, 190-197.*

[Siegmund, 1978] Siegmund, D. (1978). Estimation following sequential tests. *Biometrika 65, 341-349.*

[Siegmund, 1985] Siegmund, D. (1985). Sequential analysis. *Springer-Verlag, New York.*

[Slud and Wei, 1982] Slud, E. and Wei, L.-J. (1982). Two-sample repeated significance tests based on the modified wilcoxon statistic. *J. Amer. Statist. Assoc. 77, 862-868.*

[Tsiatis et al., 1984] Tsiatis, A., Rosner, G., and Mehta, C. (1984). Exact confidence intervals following a group sequential test. *Biomertrics 40, 797-803.*

[Wald, 1948] Wald, A. (1948). Sequential analysis. *New York: Wiley.*

[Wald and Wolfowitz, 1948] Wald, A. and Wolfowitz, J. (1948). Optimum character of the sequential probability ratio test. *Ann. Math. Statist. 19, 326-339.*

[Whitehead, 1986] Whitehead, J. (1986). On the bias of maximum likelihood estimation following a sequential test. *Biometrika 73, 573-581.*