SUPPLEMENTARY MATERIAL

Table 7: Power, expected event size, variance, squared bias, and mean squared error of $\hat{\theta}$ from the BSD4TEO, when θ_0 is set to log(1).

		BSD4TEO				
$\exp(\theta)$		Power	Size	$\text{Var}(\hat{\theta})$	bias^2	MSE
0.5	OF	1.000	111	0.0004	0.3266	0.3270
0.6	OF	0.986	142	0.0003	0.1616	0.1620
0.7	OF	0.862	171	0.0006	0.0696	0.0702
0.8	OF	0.532	191	0.0012	0.0244	0.0257
0.9	OF	0.182	200	0.0017	0.0056	0.0072
0.5	Pocock	1.000	74	0.0004	0.3702	0.3706
0.6	Pocock	0.976	106	0.0004	0.1805	0.1809
0.7	Pocock	0.805	134	0.0005	0.0758	0.0763
0.8	Pocock	0.459	148	0.0010	0.0256	0.0267
0.9	Pocock	0.164	142	0.0016	0.0056	0.0072
0.5	Uniform	1.000	78	0.0005	0.3655	0.3660
0.6	Uniform	0.980	111	0.0004	0.1779	0.1782
0.7	Uniform	0.826	141	0.0005	0.0749	0.0753
0.8	Uniform	0.482	157	0.0010	0.0255	0.0265
0.9	Uniform	0.167	156	0.0016	0.0056	0.0072

OF: O'Brien–Fleming.

Table 7 presents the simulation results of the BSD4TEO when $\theta_0 = \log(1)$, while all other assumptions are kept as in Section 4. The results in Table 7 are quite similar to those in Table 4. In addition, Table 8 shows the results when θ_0 is set to log(0.5), *i.e.* prior is strongly in favor of the treatment group. The squared bias of $\hat{\theta}$ from the BSD4TEO becomes larger as the prior setting, $\theta_0 = \log(0.5)$, is further away from the true θ.

Table 8: Power, expected event size, variance, squared bias, and mean squared error of $\hat{\theta}$ from the BSD4TEO, when θ_0 is set to $\log(0.5)$.

		BSD4TEO				
$\exp(\theta)$		Power	Size	$\text{Var}(\hat{\theta})$	bias^2	MSE
0.5	OF	1.000	110	0.0011	$< .0001$	0.0011
0.6	OF	0.989	141	0.0019	0.0204	0.0223
0.7	OF	0.863	172	0.0033	0.0619	0.0652
0.8	OF	0.530	193	0.0037	0.1082	0.1118
0.9	OF	0.203	203	0.0029	0.1583	0.1613
0.5	Pocock	1.000	76	0.0008	$< .0001$	0.0008
0.6	Pocock	0.979	105	0.0018	0.0228	0.0247
0.7	Pocock	0.805	135	0.0040	0.0681	0.0721
0.8	Pocock	0.448	145	0.0051	0.1154	0.1205
0.9	Pocock	0.168	147	0.0042	0.1634	0.1676
0.5	Uniform	1.000	79	0.0008	$< .0001$	0.0008
0.6	Uniform	0.985	111	0.0019	0.0225	0.0244
0.7	Uniform	0.831	142	0.0039	0.0672	0.0711
0.8	Uniform	0.488	159	0.0049	0.1142	0.1191
0.9	Uniform	0.186	166	0.0040	0.1622	0.1661

OF: O'Brien–Fleming.

Table 9 presents the results for applying the BSD4TEO on the CGD dataset when using a smaller τ, 0.1. The stopping times of the trial are the same as those from Table 6 for different alpha spending functions. We notice that when using the smaller τ, the posterior distributions depend less on the prior information. Therefore, BF_{01} calculated under different θ_0 are close to each other, when compared to those from $\tau = 1$ in Table 6.

Table 9: The required event size, stopping time, Bayes factor, and estimated log hazard ratio for the BSD4TEO application on CGD data set, using different θ_0 and alpha spending functions, when τ is set to 0.1.

$\exp(\theta_0)$		n	d	n_1	d_1	n_0	d_0	t_{stop}	BF_{01}	$\hat{\theta}$
0.3349	O'Brien–Fleming	128	30	63	7	65	23	333	0.0032	-1.433
	Pocock	128	15	63	3	65	12	211	0.0434	-1.501
	Uniform	128	15	63	3	65	12	211	0.0434	-1.501
0.5	O'Brien–Fleming	128	30	63	7	65	23	333	0.0034	-1.374
	Pocock	128	15	63	3	65	12	211	0.0464	-1.392
	Uniform	128	15	63	3	65	12	211	0.0464	-1.392
1	O'Brien–Fleming	128	30	63	7	65	23	333	0.0041	-1.275
	Pocock	128	15	63	3	65	12	211	0.0564	-1.218
	Uniform	128	15	63	3	65	12	211	0.0564	-1.218

BSD4TEO.R: R code to implement Algorithm 1 and Algorithm 2.

```
require("BRugs")
require("survival")

# likelihood function
Bernlik <- function(theta, d, y0, y1) {
  #theta:log hazard ratio of the treatment over the control
  #d:event observed, ordered in times
  #1=event observed in the treatment group, 0=event observed in the control
      group
  #y0:the number of subjects at risk in the control group, ordered in times
  #y1:the number of subjects at risk in the treatment group, ordered in
      times
  lik <- 1
  for(i in 1:length(d)) {
    p <- exp(theta)*y1[i] / (y0[i]+exp(theta)*y1[i])
    lik <- lik*dbinom(d[i], 1, p)
  }
  return (lik)
}

# OBrien-Fleming alpha spending function
OF.asf <- function(inf.frac, alpha=0.05) {
  #inf.frac:vector of infomration fraction
  #alpha:overall type I error rate
  2 - 2*pnorm( qnorm(1-alpha/2) / sqrt(inf.frac) )
}

# calculate critical values using alpha spending function
asf2cv <- function(mat, fun.asf, inf.frac, alpha=0.05) {
  #mat:a M*J matrix of Bayes factor in favor of H0
  #M=the number of simulated trials, J=the number of interim analyses
  #fun.asf:the alpha spending function to be used
  #inf.frac:vector of infomration fraction
  #alpha:overall type I error rate
  asf1 <- fun.asf(inf.frac, alpha)
  matBF.temp <- mat
  cv.P <- quantile(matBF.temp[, 1], asf1[1], names=F)
  for (k in 2:length(inf.frac)) {
    matBF.temp <- matBF.temp[which(matBF.temp[ , (k-1)]>= cv.P[k-1]), ]
    cv.P <- c(cv.P, quantile(matBF.temp[ ,k], (asf1[k]-asf1[k-1]), na.rm=T,
        names=F))
  }
  return (cv.P)
}

# generating survival data
surv_gen <- function(ncon, ntrt, betas=0, baseh="exponential", scale=3,
    shape=5){
  #ncon:the number of patients in control group
  #ntrt:the number of patients in treatment group
  #betas:log hazard ratio
  #baseh:baseline hazard function, exponential, weibull or Gomptez
  X <- rep(c(0,1), c(ncon, ntrt))
  U <- runif(ncon+ntrt, min=0, max=1)
  ifelse(baseh=="exponential", time<- t(-1 * log(U)/(scale * exp(t(betas)
      %*% X))),
      ifelse(baseh=="weibull", time<- t((-1 * log(U)/(scale * exp(
          t(betas) %*% X)))^(1/shape)),
              # Gomptez
              time<- t((1/shape) * log( 1 - ((shape * log(U))/(scale *
                  exp(t(betas) %*% X)))))))
  return(as.data.frame(cbind(time=time[, 1], trt=X)))
}

# call openbugs
bugs.model <- function(model, samples, data=list(), chainLength=5000,
    burnin=0.10, n.chains=3) {
  #model:the model as a string
  #samples:the variable to be monitored
```

```
  #data:datalist
  writeLines(model, con="model.txt")
  modelCheck("model.txt")
  if (length(data)>0)   modelData(bugsData(data))
  modelCompile(n.chains)
  modelGenInits()
  modelUpdate(chainLength*burnin)
  samplesSet(samples)
  modelUpdate(chainLength)
}

## calculate critical value for the BSD4TE0
CriValue.TE <- function(ncon, ntrt, IA.E, st.fun, st.args,
                        baseh.E="exponential", scale.E, shape.E,
                        ce.fun, ce.args,
                        fun.asf, alpha=0.05, maxit=1000,
                        theta0=0, sigma0=0.1, tau=1,
                        chainLength=5000, burnin=0.10, n.chains=3) {
  #ncon:the number of patients in control group
  #ntrt:the number of patients in treatment group
  #IA.E:a vector of preset number of events observed for each interim
  #st.fun:the function to generate entry times with the first argument
      n=number of obs
  #st.args:optional arguments to st.fun
  #baseh.E:baseline hazard function, exponential, weibull or Gomptez
  #ce.fun:the function to generate censoring times with the first argument
      n=number of obs
  #ce.args:optional arguments to ce.fun
  #fun.asf:the alpha spending function to be used
  #maxit:number of simulated trials
  #theta0:the mean of prior distribution
  #sigma0, tau:sigma0^2/tau is the variance of prior distribution
  prec.theta0 <- tau/sigma0^2
  Nia <- length(IA.E)

  mat.BF01 <- NULL
  #simulate maxit trials
  for (m in 1:maxit) {
    ##generate data
    time.start <- round( do.call(st.fun, c(list(n=ncon+ntrt), st.args)) )
    time.event <- surv_gen(ncon, ntrt, 0, baseh.E, scale.E, shape.E)
    time.event$time <- round(time.event$time)
    time.censor <- round( do.call(ce.fun, c(list(n=ncon+ntrt), ce.args)) )
    ##if follow-up until all subjects have event/cencored
    time.gen <- data.frame(time=pmin(time.event$time, time.censor),
        trt=time.event$trt, censor=as.numeric(time.event$time <=
        time.censor), tst=time.start)
    time.gen$tend <- time.gen$tst + time.gen$time
    #order by time of subjects leaving the trial
    time.gen <- time.gen[order(time.gen$tend),]
    BF01.temp <- NULL
    #for each trial
    for (j in 1:Nia) {
      dat<-time.gen
      ##Calculate time when the preset number of events are observed
      if(sum(dat$censor) >= IA.E[j]) {
        IA.end <- dat[dat$censor==1,][IA.E[j],]$tend
        dat <- dat[dat$tst <= IA.end, ]
        for (i in 1:nrow(dat) ) {
          if (dat$tend[i] > IA.end) {
            dat$time[i] <- IA.end-dat$tst[i]
            dat$censor[i] <- 0
            dat$tend[i] <- IA.end
          }
        }
      }
      #prepare datalist
      s <- summary( survfit( Surv(time, censor)~1, data=dat) )
      d <- NULL
      y0 <- NULL
      y1 <- NULL
```

```
      k <- 1
      while (k <= length(s$n.event)) {
        if (s$n.event[k]==1) {
          d <- c(d, dat$trt[dat$time==s$time[k] & dat$censor==1])
          y1 <- c(y1, sum( dat$trt[dat$time >= s$time[k]] ) )
          y0 <- c(y0, s$n.risk[k]- sum( dat$trt[dat$time >= s$time[k]] ) )
          k <- k + 1
        } else {
          d <- c(d, dat$trt[dat$time==s$time[k] & dat$censor==1])
          y1 <- c(y1, rep( sum( dat$trt[dat$time >= s$time[k]] ),
              s$n.event[k]) )
          y0 <- c(y0, rep( s$n.risk[k]-sum( dat$trt[dat$time >= s$time[k]]
              ), s$n.event[k]) )
          k <- k + 1
        }
      }
      #calculate Bayes factor BF01
      ##call openbugs
      modelString <- paste("
        model {
          for(i in 1:N) {
            d[i] ~ dbern(p[i])
            p[i] <- exp(theta)*y1[i]/( y0[i]+exp(theta)*y1[i])
        }",
      "theta ~ dnorm(", theta0, ",", prec.theta0, ")",
      "}", collapse=",")
      data.list <- list(
        d = d,
        y1 = y1,
        y0 = y0,
        N = length(d))
      bugs.model(modelString, samples=c("theta"), data=data.list,
                 chainLength, burnin, n.chains)
      rs.temp <- samplesSample(c("theta"))
      ml.temp <- 1/mean(1/Bernlik(rs.temp[rs.temp<0], d, y0, y1))
      BF01.temp <- c(BF01.temp, Bernlik(0, d, y0, y1)/ml.temp )
    }
    mat.BF01 <- rbind(mat.BF01, BF01.temp)
  }
  #calculate critical values
  cv <- asf2cv(mat.BF01, fun.asf, IA.E/IA.E[Nia], alpha)
  return (cv)
}

## calculate power for the BSD4TEO
PWR.TE <- function(ncon, ntrt, IA.E, log.hr, cv,
                   st.fun, st.args,
                   baseh.E="exponential", scale.E=3, shape.E=5,
                   ce.fun, ce.args,
                   maxit=1000, theta0=0, sigma0=0.1, tau=1,
                   chainLength=5000, burnin=0.10, n.chains=3) {
  #ncon: number of patients in control group
  #ntrt: number of patients in treatment group
  #IA.E: a vector of preset number of events observed for each interim
  #log.hr: log hazard ratio
  #cv: a vector of critical values
  #st.fun: the function to generate entry times with the first argument
      n=number of obs
  #st.args: optional arguments to st.fun
  #baseh: baseline hazard function, exponential, weibull or Gomptez
  #ce.fun: the function to generate censoring times with the first argument
      n=number of obs
  #ce.args: optional arguments to ce.fun
  #maxit: number of simulated trials
  #theta0: the mean of prior distribution
  #sigma0, tau: sigma0^2/tau is the variance of prior distribution
  prec.theta0 <- tau/sigma0^2
  Nia <- length(IA.E)

  mat.BF01 <- NULL
  #simulate maxit trials
```

```
for (m in 1:maxit) {
  ##generate data
  time.start <- round( do.call(st.fun, c(list(n=ncon+ntrt), st.args)) )
  time.event <- surv_gen(ncon, ntrt, log.hr, baseh.E, scale.E, shape.E)
  time.event$time <- round(time.event$time)
  time.censor <- round( do.call(ce.fun, c(list(n=ncon+ntrt), ce.args)) )
  ##if follow-up until all subjects have event/cencored
  time.gen <- data.frame(time=pmin(time.event$time, time.censor),
     trt=time.event$trt, censor=as.numeric(time.event$time <=
     time.censor), tst=time.start)
  time.gen$tend <- time.gen$tst + time.gen$time
  #order by time of subjects leaving the trial
  time.gen <- time.gen[order(time.gen$tend),]
  BF01.temp <- NULL
  #for each trial
  for (j in 1:Nia) {
    dat<-time.gen
    ##Calculate time when the preset number of events are observed
    if(sum(dat$censor) >= IA.E[j]) {
      IA.end <- dat[dat$censor==1,][IA.E[j],]$tend
      dat <- dat[dat$tst <= IA.end, ]
      for (i in 1:nrow(dat) ) {
        if (dat$tend[i] > IA.end) {
          dat$time[i] <- IA.end-dat$tst[i]
          dat$censor[i] <- 0
          dat$tend[i] <- IA.end
        }
      }
    }
    #prepare datalist
    s <- summary( survfit( Surv(time, censor)~1, data=dat) )
    d <- NULL
    y0 <- NULL
    y1 <- NULL
    k <- 1
    while (k <= length(s$n.event)) {
      if (s$n.event[k]==1) {
        d <- c(d, dat$trt[dat$time==s$time[k] & dat$censor==1])
        y1 <- c(y1, sum( dat$trt[dat$time >= s$time[k]] ) )
        y0 <- c(y0, s$n.risk[k]- sum( dat$trt[dat$time >= s$time[k]] ) )
        k <- k + 1
      } else {
        d <- c(d, dat$trt[dat$time==s$time[k] & dat$censor==1])
        y1 <- c(y1, rep( sum( dat$trt[dat$time >= s$time[k]] ),
            s$n.event[k]) )
        y0 <- c(y0, rep( s$n.risk[k]-sum( dat$trt[dat$time >= s$time[k]]
            ), s$n.event[k]) )
        k <- k + 1
      }
    }
    # calculate Bayes factor BF01
    ## call openbugs
    modelString <- paste("
      model {
        for(i in 1:N) {
          d[i] ~ dbern(p[i])
          p[i] <- exp(theta)*y1[i]/( y0[i]+exp(theta)*y1[i])
        }",
    "theta ~ dnorm(", theta0, ",", prec.theta0, ")",
    "}", collapse=",")
    data.list <- list(
      d = d,
      y1 = y1,
      y0 = y0,
      N = length(d))
    bugs.model(modelString, samples=c("theta"), data=data.list,
               chainLength, burnin, n.chains)
    rs.temp <- samplesSample(c("theta"))
    ml.temp <- 1/mean(1/Bernlik(rs.temp[rs.temp<0], d, y0, y1))
    BF01.temp <- c(BF01.temp, Bernlik(0, d, y0, y1)/ml.temp )
    }
```

```
    mat.BF01 <- rbind(mat.BF01, BF01.temp)
  }
  #calculate power
  id.temp <- apply(mat.BF01, 1, function(x, y) which(x<y)[1], "y"=cv)
  pwr <- sum(!is.na(id.temp))/length(id.temp)
  return (pwr)
}


## Example: calculation of critical values for application on CGD dataset
## using prior theta_0=log(0.5), sigma_0=0.3, tau=0.1
## number of replicates=1000
cv.OF <- CriValue.TE(ncon=65, ntrt=63, IA.E=c(15, 30, 44),
            st.fun=runif, st.args=list(min=0, max=205),
            baseh.E="exponential", scale.E=1/457, shape.E=1,
            ce.fun=rexp, ce.args=list(rate=1/391),
            fun.asf=OF.asf, alpha=0.05, maxit=1000,
            theta0=log(0.5), sigma0=0.3, tau=0.1,
            chainLength=5000, burnin=0.10, n.chains=1)

## Example: power calculation for the above settings when hazard ratio is
    0.3349
PWR.TE(ncon=65, ntrt=63, IA.E=c(15, 30, 44), log.hr=log(0.3349), cv=cv.OF,
          st.fun=runif, st.args=list(min=0, max=205),
          baseh.E="exponential", scale.E=1/457, shape.E=1,
          ce.fun=rexp, ce.args=list(rate=1/391),
          maxit=1000, theta0=log(0.5), sigma0=0.3, tau=0.1,
          chainLength=5000, burnin=0.10, n.chains=1)
```