

SIMULATION-BASED SAMPLE SIZE CALCULATOR FOR THE RPPD

L Abrahamyan, CS Li, J Beyene, AR Willan, BM Feldman

Lusine Abrahamyan MD MPH^{1,3}, Chuan Silvia Li^{1,4}, Joseph Beyene MSc PhD^{1,3,5}, Andrew R. Willan PhD^{1,5}, Brian M. Feldman MD MSc FRCPC^{1,2,3,5}

¹Child Health Evaluative Sciences, Research Institute, The Hospital for Sick Children, ²Departments of Paediatrics, ³Health Policy Management & Evaluation, ⁴Pharmacology, ⁵Dalla Lana School of Public Health, University of Toronto, Toronto, Ontario, Canada

This discrete time, object-oriented simulation program is developed for the investigation of the statistical efficiency (power) of the Randomized Placebo-Phase Design (RPPD) and for parallel group RCTs with time-to-event outcomes, using R statistical software (*Version 2.6.1, The R Foundation for Statistical Computing*). The program calculates the power of trials for different sample sizes when the response times to experimental or standard treatment follow exponential, Weibull or lognormal distributions. Appendix A presents the program flow chart. Appendix B is the program code with clinical trial design input parameters that can be modified by the user. Appendix C is the source code ('RPPD.R') for the simulation function that should be saved in the same directory as the program code (Appendix B).

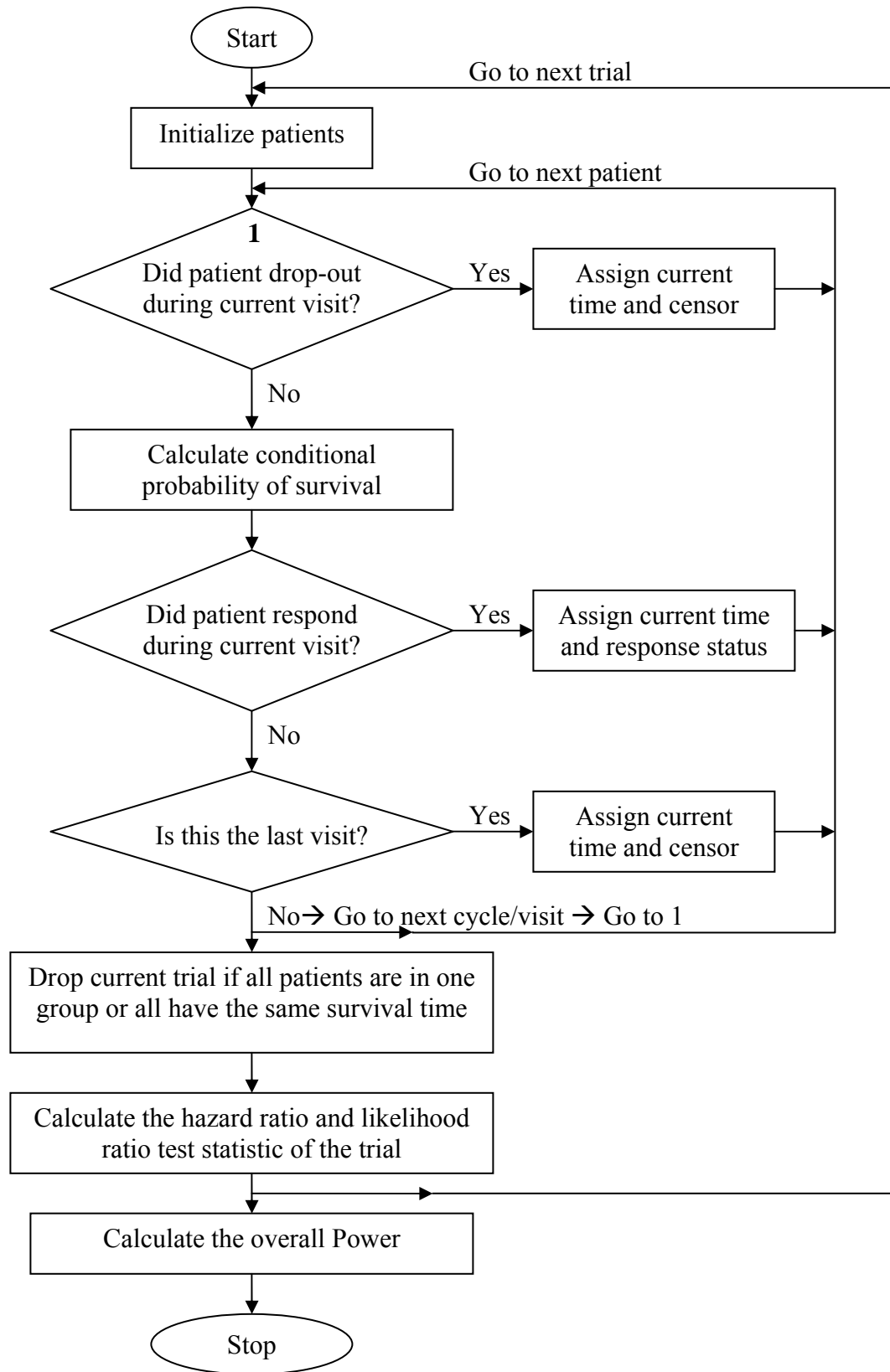
Directions: In order to run the simulation and calculate study power, input the corresponding parameters included in the “***CLINICAL TRIAL DESIGN MODIFIABLE PARAMETERS***” section of Appendix B and submit the code. For large sample sizes (total $n \geq 50$) and for a large number of simulations (≥ 1000 iterations) you may consider using a high performance computer.

Notes:

More information about 'R.oo' package for object-oriented simulations is available at: <http://www1.maths.lth.se/help/R/R.oo/>

The description of the Randomized Placebo-Phase Design (RPPD) is available in “Feldman B, Wang E, Willan A, Szalai JP. The randomized placebo-phase design for clinical trials. *J Clin Epidemiol* 2001;54(6):550-7 ”.

Appendix A. Simulation Program Flow Chart



Appendix B. Simulation Program Code

```
#=====PROGRAM START=====#
library(survival) # load the R package for survival analysis
library(R.oo) # load the R package for object-oriented simulations
source("RPPD.r") # source 'RPPD.R' function used in simulation
patients <- list() # define patient object as a list (Global Variable)
EXPONENTIAL = 1
WEIBULL = 2
LOGNORMAL = 3
EXP_PHASE = 1
PLACEBO_PHASE = 0
FILE = 1
SCREEN = 2
RPPD = 1
PARALLEL = 2

#-----CLINICAL TRIAL DESIGN MODIFIABLE PARAMETERS-----#
simulation = PARALLEL # the trial design is 'PARALLEL' OR 'RPPD'
placebo_distribution = EXPONENTIAL # placebo treatment distribution is 'EXPONENTIAL' or 'WEIBULL' or
'LOGNORMAL'
exp_distribution = EXPONENTIAL # experimental treatment distribution is 'EXPONENTIAL' or 'WEIBULL' or
'LOGNORMAL'
MAX_PLACEBO_LENGTH = 60 # maximum placebo phase duration for the RPPD design in days
ACCRUAL_PHASE = 90 # accrual time in days
MAX_EXP_LENGTH = 275 # maximum duration for the experimental treatment in days
CYCLE_TIME = 14 # interval duration for patient follow-up visits in days
NUM_PATIENT = 8 # total number of patients in the study
num_trials = 100 # number of simulated trials/iterations
alpha=0.05 # Type I error rate

exp_drop_out = 0.10 # patient drop-out probability (random) for the experimental group
control_drop_out = 0.10 # patient drop-out probability (random) for the control group
rppd_accrual_prob = 1.0 # patient accrual probability in the RPPD design
parallel_accrual_prob = 1.0 # patient accrual probability in the parallel group design

baseline_lambda = 0.00195 # baseline (daily) lambda for the exponential distribution
treatment_lambda = 0.0165 # experimental treatment lambda (daily) for the exponential distribution

baseline_d_shape = 3 # baseline shape for the Weibull or mean logT for the lognormal distribution
baseline_d_scale = 0.021 # baseline scale for the Weibull or SD of logT for the lognormal distribution

treatment_d_shape = 3.74 # experimental treatment shape for the Weibull or mean logT for the lognormal distribution
treatment_d_scale = 0.95 # experimental scale for the Weibull or SD of logT for the lognormal distribution

output_option = SCREEN # display the results on the 'SCREEN' or save in the 'FILE'
output_file = "output.txt" # specify the output file name if 'output_option = FILE'

#=====SEED OPTIONS=====#
seed = Sys.time() # set the seed equal to the system time
n_seed = as.numeric(seed) # convert the seed to the numeric value
output("Seed: ",n_seed, "\n") # print the seed
set.seed(as.numeric(seed)) # set the randomization seed equal to system time

#-----PLACEBO PHASE DURATION DEPENDING ON THE RCT DESIGN-----#
max_exp_cycle= MAX_EXP_LENGTH%%CYCLE_TIME #calculate the number of cycles in the experimental phase
```

```
if (simulation == RPPD) {
  max_placebo_cycle= MAX_PLACEBO_LENGTH%%CYCLE_TIME # calculate the number of cycles/patient visits in the
  placebo-phase of the RPPD
  accrual_prob = rppd_accrual_prob # assign accrual probability for the RPPD
} else if (simulation == PARALLEL) {
  MAX_PLACEBO_LENGTH = MAX_EXP_LENGTH + 1
  max_placebo_cycle= MAX_PLACEBO_LENGTH%%CYCLE_TIME # calculate the number of cycles/patient visits in the
  placebo group of the parallel group RCT
  accrual_prob = parallel_accrual_prob # assign accrual probability for the parallel group RCT
}

start_simulation () # run the simulation

#=====PROGRAM END=====#
```

Appendix C. 'RPPD.R' function

```
#=====PROGRAM START=====#

#=====FUNCTION TO DISPLAY RESULTS/PARAMETERS=====#
output <- function(msg1=", msg2=", msg3=", msg4=", msg5=", msg6=", msg7 = ", msg8 = ", msg9 = ", msg10 = ", msg11 = ",
msg12 = ") {
  if (output_option == FILE)
    cat(paste(msg1, msg2, msg3, msg4, msg5, msg6, msg7, msg8, msg9, msg10, msg11, msg12, sep = ""), file=output_file,
append = TRUE)
  else if (output_option == SCREEN)
    cat(paste(msg1, msg2, msg3, msg4, msg5, msg6, msg7, msg8, msg9, msg10, msg11, msg12, sep = ""))
}

#=====DEFINE CHARACTERISTICS OF THE "Patient" - OBJECT =====#
setConstructorS3("Patient", function(patientID, lambda, d_scale, d_shape, placebo_phase_length, status, survival_time,
probSurv, accrual_time, drop_out, is_enrolled, phase) {
  if (missing(patientID)) patientID <- NA;
  if (missing(lambda)) lambda <- NA;
  if (missing(d_scale)) d_scale <- NA;
  if (missing(d_shape)) d_shape <- NA;
  if (missing(probSurv)) probSurv <- NA;
  if (missing(placebo_phase_length)) placebo_phase_length <- NA;
  if (missing(status)) status <- NA;
  if (missing(survival_time)) survival_time <- NA;
  if (missing(accrual_time)) accrual_time <- NA;
  if (missing(drop_out)) drop_out <- NA;
  if (missing(is_enrolled)) is_enrolled <- NA;
  if (missing(phase)) phase <- NA;
  extend(Object(), "Patient",
    .patientID=patientID,
    .lambda=lambda,
    .probSurv=probSurv,
    .placebo_phase_length=placebo_phase_length,
    .d_shape = d_shape,
    .d_scale= d_scale,
    .status=status,
    .survival_time=survival_time,
    .accrual_time=accrual_time,
    .drop_out=drop_out,
    .is_enrolled=is_enrolled,
    .phase=phase
  )
})

#=====SIMULATE ALL PATIENTS FOR ALL TRIALS=====#
runTrial <- function() {
  lrt_list <- list() # store the likelihood Ratio Test (LRT) statistic from each trial in a list
  realTrialNum = 1 # set the real trial number as one

  for (j in 1:num_trials) { # initialize trials
    if ((realTrialNum == 1) || (realTrialNum %% 10 == 0) || (j==num_trials)) # print the trial number for every i-th iteration to
observe simulation progress
      output("\nTrial #", realTrialNum, "-----\n")

    allInOneGroup = TRUE # set 'all patients are randomized to one group' as true

    for(i in 1:NUM_PATIENT) { # initialize patients
      patients[[i]] <<- Patient()
    }
  }
}
```

```

patients[[i]]$.patientID <<- i
patients[[i]]$.is_enrolled <<- ifelse(runif(1) < accrual_prob, 1, 0) #assign accrual probability
patients[[i]]$.placebo_phase_length <<- ifelse(runif(1) < 0.5, 0, MAX_PLACEBO_LENGTH) # assign placebo phase
duration
patients[[i]]$.phase <<- ifelse(patients[[i]]$.placebo_phase_length ==0, EXP_PHASE, PLACEBO_PHASE) # assign
treatment arms

if (patients[[i]]$.is_enrolled == 1) { # if the patient is enrolled
  patients[[i]]$.status <<- 0 # assign initial values for status, survival time and survival probability
  patients[[i]]$.survival_time <<- 0
  patients[[i]]$.probSurv <<- 1

#Check if all patients are in the same group
if (allInOneGroup == TRUE && i!=1 && patients[[i]]$.placebo_phase_length!=patients[[i-1]]$.placebo_phase_length)
  allInOneGroup = FALSE
  # Assign distribution parameters based on treatment allocation at enrollment
  patients[[i]]$.lambda <<- ifelse(patients[[i]]$.placebo_phase_length==0, treatment_lambda, baseline_lambda)
  patients[[i]]$.d_scale <<- ifelse(patients[[i]]$.placebo_phase_length==0, treatment_d_scale, baseline_d_scale)
  patients[[i]]$.d_shape <<- ifelse(patients[[i]]$.placebo_phase_length==0, treatment_d_shape, baseline_d_shape)
  patients[[i]]$.accrual_time <<- runif(1, 0, ACCRUAL_PHASE) # assign accrual time
  patients[[i]]$.drop_out <<- 0 # assign initial drop-out status
}
}

if (allInOneGroup ==FALSE) { # if patients are randomized to different arms
runPhase() # simulate current trial
allsamesurvivaltime = TRUE # set 'all patients have the same survival time' as true
for(i in 2:NUM_PATIENT) {
  if (patients[[i]]$.survival_time!= patients[[i-1]]$.survival_time){ # compare survival times
    allsamesurvivaltime = FALSE
    break
  }
}

if (allsamesurvivaltime == FALSE ) { # if all patients do not have the same survival time
  lrt_list[[realTrialNum]] = fitCox(realTrialNum) # add the LRT from the current trial to the list
  realTrialNum = realTrialNum+1 # add the trial to real trial number after removing trials that are 'allInOneGROUP' or
have 'allsamesurvival'
}
}
}
output("\n\n~~~~~\n")

# Calculate the Power of the simulation scenario after all iterations
if (simulation == RPPD) {
  lrt_list_count = 0

  for (k in 1:(realTrialNum-1)) {
    # Check if the curent trial has a hazard ratio < 1 (observed difference) and LRT greater than the critical value of the Chi
square test for 1 df and one-sided alpha
    if ((lrt_list[[k]][[2]] < 1) && (lrt_list[[k]][[1]] >= qchisq(1-2*alpha, 1))){
      lrt_list_count = lrt_list_count + 1 # add to LRT list if condition satisfied
    }
  }
  coxPowerRPPD = lrt_list_count / (realTrialNum-1) # calculate the power of the simulation scenario for the RPPD
  output("RPPD Cox Power: ", coxPowerRPPD, "\n")
} else if (simulation == PARALLEL){
  lrt_list_count = 0

```

```

for (k in 1:(realTrialNum-1)) {
  # Check if the current trial has a hazard ratio < 1 (observed difference) and LRT greater than the critical value of the
  Chi square test for 1 df and one or two-sided alpha
  if ((lrt_list[[k]][[2]] < 1) && (lrt_list[[k]][[1]] >= qchisq(1-alpha, 1))) {
    lrt_list_count = lrt_list_count + 1 # add to LRT list if condition satisfied
  }
}
coxPowerParallel = lrt_list_count/(realTrialNum-1) # calculate the power of the simulation scenario for the parallel group
RCT
output("Parallel Cox Power: ", coxPowerParallel, "\n")
}
}

```

#####SIMULATE PATIENTS FOR THE CURRENT TRIAL AND DECIDE THE STATUS#####

```

runPhase <- function(){
for (j in 1:NUM_PATIENT) {
  if (patients[[j]]$.is_enrolled == 1) { # for all enrolled patients
    for(i in 1:max_exp_cycle) { # initialize cycles/visits
      if (i == 1) {
        patients[[j]]$.drop_out <- 0 # assign a drop-out probability of 0 at the entry
      } else {
        if (patients[[j]]$.placebo_phase_length == 0) { # assign a drop-out probability depending on phase/group assignment
          drop_out = exp_drop_out
        } else {
          drop_out = control_drop_out
        }
        patients[[j]]$.drop_out <- ifelse(runif(1) < drop_out, 1, 0) # calculate the drop-out probability
      }

      if (patients[[j]]$.drop_out == 0) { # if the patient did not drop-out at the current cycle
        findprobSurv(i, j) # calculate the probability of survival
        patients[[j]]$.status <- ifelse(runif(1) < patients[[j]]$.probSurv, 0, 1) # define the response status

        if (patients[[j]]$.status == 1) { # if the patient status is 1 ('patient responded')
          patients[[j]]$.survival_time <- CYCLE_TIME*i # calculate the survival time
          break # go to next patient
        }

        if (i==max_exp_cycle && patients[[j]]$.status==0) { # if the trial ends and the status is 0
          patients[[j]]$.survival_time <- i * CYCLE_TIME # calculate the survival time
          patients[[j]]$.drop_out <- 1 # censor the patient
          break # go to next patient
        }
      }

      else if (patients[[j]]$.drop_out == 1) { # if the patient is censored/drops out from the current cycle
        patients[[j]]$.survival_time <- CYCLE_TIME*i # calculate the survival time
        break # go to next patient
      }
    }
  }
}
}
}
}

```

#####CALCULATE THE PROBABILITY OF SURVIVAL FOR EACH PATIENT#####

```

findprobSurv <- function(current_cycle_num, patientIndex) {
  # Check if the patient was assigned to placebo and finished the placebo-phase
  if (current_cycle_num == (patients[[patientIndex]]$.placebo_phase_length%% CYCLE_TIME + 1) &&
  patients[[patientIndex]]$.placebo_phase_length != 0) {
    patients[[patientIndex]]$.phase <- EXP_PHASE # change to experimental phase (happens only in the RPPD)
  }
}

```

```

# Change the distribution parameters from baseline to treatment specific
if (exp_distribution == EXPONENTIAL)
  patients[[patientIndex]]$.lambda <<- treatment_lambda
else if (exp_distribution == WEIBULL || exp_distribution == LOGNORMAL) {
  patients[[patientIndex]]$.d_scale <<- treatment_d_scale
  patients[[patientIndex]]$.d_shape <<- treatment_d_shape
}
}

# Assign the survival distribution to each arm
if (patients[[patientIndex]]$.phase == EXP_PHASE)
  distribution_model = exp_distribution
else if (patients[[patientIndex]]$.phase == PLACEBO_PHASE)
  distribution_model = placebo_distribution

# Calculate the conditional probability of survival using S(t+1)/S(t) formula according to distribution
# For exponential distribution, S(T) = exp(-lambda*t) where lambda is the daily hazard and t is the time
if (distribution_model == EXPONENTIAL)
  patients[[patientIndex]]$.probSurv <<- exp((-1)*(patients[[patientIndex]]$.lambda*CYCLE_TIME))
# For Weibull distribution, S(t) = exp[-(scale*t)^shape]
else if (distribution_model == WEIBULL)
  patients[[patientIndex]]$.probSurv <<- (exp(-
(patients[[patientIndex]]$.d_scale*((current_cycle_num+1)*CYCLE_TIME))^patients[[patientIndex]]$.d_shape))/(exp(-
(patients[[patientIndex]]$.d_scale*(current_cycle_num*CYCLE_TIME))^patients[[patientIndex]]$.d_shape))
# For lognormal distribution, S(t) = 1-((log(exp(-mean)*t))/SD) where G is the cumulative distribution function of a standard
normal variable
else if (distribution_model == LOGNORMAL) {
  a<-exp(-patients[[patientIndex]]$.d_shape)
  patients[[patientIndex]]$.probSurv <<- (1-
pnorm(log(a*(current_cycle_num+1)*CYCLE_TIME)/patients[[patientIndex]]$.d_scale))/(1-
pnorm(log(a*(current_cycle_num)*CYCLE_TIME)/patients[[patientIndex]]$.d_scale))
}
}

#=====STORE PATIENT INFORMATION AND CALCULATE THE HR & LRT FOR EACH TRIAL=====#
fitCox <- function(realTrialNum) {
  # Create an empty data frame
  survival.data <- data.frame(time=numeric(length(patients)), status=numeric(length(patients)),
group=numeric(length(patients)))
  counter = 1
  # Store the information of each enrolled patient into the data frame
  for(i in 1:length(patients)) {
    if (patients[[i]]$.is_enrolled == 1) {
      survival.data [counter,] <- c(patients[[i]]$.survival_time, patients[[i]]$.status,patients[[i]]$.placebo_phase_length)
      counter = counter + 1
    }
  }
  # Fit the Cox proportional hazard model using 'coxph' function from R
  coxfit <- coxph(Surv(time, status)~group, data=survival.data)
  HazardRatio <- exp(coxfit$coef) # calculate the Hazard ratio
  loglik <- coxfit$loglik # obtain the log-likelihood statistic from the model
  warning("Convergence problem in 'loglik' ", call. = FALSE)
  lrt <- -2*(loglik[1]-loglik[2]) # calculate the LRT statistic
  retval<-data.frame(lrt,HazardRatio) # store the values to perform power calculation
  return(retval)
}

start_simulation <- function () { # start the simulation of the current scenario

# =====DISPLAY PARAMETERS USED IN CURRENT SIMULATION=====#
output("\n~~~~~\n")

```

```

if (simulation == RPPD) {
  output("RPPD Design\n")
} else if (simulation == PARALLEL){
  output("Parallel Design\n")}
if (exp_distribution == EXPONENTIAL) {
  output("Distribution for experimental phase/group: ")
  output("Exponential Distribution\n")
  output("Daily Treatment Lambda: ", treatment_lambda, "\n")
} else if (exp_distribution == WEIBULL) {
  output("Distribution for experimental phase/group: ")
  output("Weibull Distribution\n")
  output("Shape Parameter: ", treatment_d_shape, "\n")
  output("Scale Parameter: ", treatment_d_scale, "\n")
} else if (exp_distribution == LOGNORMAL) {
  output("Distribution for experimental phase/group: ")
  output("Lognormal Distribution\n")
  output("Mean: ", treatment_d_shape, "\n")
  output("SD: ", treatment_d_scale, "\n")
}
}
if (placebo_distribution == EXPONENTIAL) {
  output("Distribution for placebo phase/group: ")
  output("Exponential Distribution\n")
  output("Daily Baseline Lambda: ", baseline_lambda, "\n")
} else if (placebo_distribution == WEIBULL) {
  output("Distribution for placebo phase/group: ")
  output("Weibull Distribution\n")
  output("Shape Parameter: ", baseline_d_shape, "\n")
  output("Scale Parameter: ", baseline_d_scale, "\n")
} else if (placebo_distribution == LOGNORMAL) {
  output("Distribution for placebo phase/group: ")
  output("Lognormal Distribution\n")
  output("Mean: ", baseline_d_shape, "\n")
  output("SD: ", baseline_d_scale, "\n")
}
}
output("Total Sample Size: ", NUM_PATIENT, "\n")
output("Number of Trials: ", num_trials, "\n")
output("~~~~~\n")
runTrial()
}
#=====PROGRAM END=====#

```