R code for simulation experiments and dose-boundary identification

Associated manuscript: Joint Identification of Efficacy and Safety Bounds in Binary Endpoint Studies

Overview

This supplement reproduces, without alteration, the three R scripts supplied with the revised manuscript. The scripts implement the baseline and higher-efficacy simulation scenarios, calculate empirical familywise error rates (FWER), simultaneous coverage diagnostics, and probabilities of correctly identifying the minimum effective dose (MinED), maximum safe dose (MaxSD), and both boundaries jointly. The scripts are written in base R.

Execution instructions

Save the three scripts shown below as separate plain-text .R files using the exact filenames specified in the table. Put all three files in the same working directory. Start a fresh R session, set the working directory, and run the scripts in this order:

```
source('phase6_simulation_verified_extended.R')
source('phase6_enhanced_verified_extended.R')
source('combine_verified_simulations_FIXED.R')
```

The first two scripts generate simulation summaries in CSV format. The third script reads these summaries and creates combined CSV tables. These outputs are produced locally by the code and are not supplied as part of this supplement.

Configuration	Specification
Simulation repetitions	10,000 per configuration
Sample size	50, 100, and 200 per group
Dose groups	Placebo/control plus three ordered active doses
Dependence structures	Negative Fréchet bound; independence; positive Fréchet bound
FM approximation	One-sided restricted-variance score tests with normal-calibrated Bonferroni critical value
Benchmark	Conservative Clopper–Pearson marginal bounds with Bonferroni adjustment
Reproducibility seeds	20261103 (baseline); 20261104 (enhanced efficacy)

Interpretive and reproducibility note

The Farrington–Manning implementation uses an approximate normal-calibrated score region; it is not an exact finite-sample nuisance-calibrated confidence region. The code distinguishes elementary FWER evaluated before sequential restrictions from stepwise FWER. The simulations use a common paired binary-outcome cohort per dose, whereas the published clinical illustration reports endpoint-specific analysis populations.

The three scripts reproduce the principal simulation runs. The original-efficacy rows of manuscript Table 6 were reported from a separately seeded follow-up simulation and are not independently reproduced by these three scripts. The distinction is retained to avoid overstating the scope of the computational supplement.

Supplementary Script S1

Baseline simulation: five scenarios

Filename: phase6_simulation_verified_extended.R

```
# VERIFIED EXTENSION: preserves the original trial-generation sequence,
# original random seed, FWER_all_elementary and FWER_stepwise definitions.
# Adds separate correct_MinED and correct_MaxSD probabilities and CIs.
# Run BOTH this and the enhanced file, then combine their outputs.
# PHASE 6: Simultaneous efficacy/safety inference for 3 ordered doses
# Base R only. Uses a COMMON PATIENT COHORT per dose in the simulations.
# This differs from the published Sakai endpoint-specific analysis populations.
# Method FM: one-sided Farrington-Manning restricted-variance score tests,
# with normal Bonferroni critical value z_(1-alpha/(2*k)).
# The score region is approximate, NOT exact nuisance-calibrated C_alpha(Z).
# Method CP: exact one-sided Clopper-Pearson binomial bounds on 2*(k+1)
# distinct marginal probabilities, Bonferroni-adjusted, then differences.
# CP provides finite-sample simultaneous coverage without endpoint independence.

set.seed(20261103)
alpha <- 0.05
delta <- 0.01
gamma <- 0.60
n_rep <- 10000L       # Change to 1000L for a fast test before a full run.
sample_sizes <- c(50L, 100L, 200L)
dependence_modes <- c('negative', 'independent', 'positive')
out_dir <- getwd()     # Files save to YOUR R working directory, not /mnt/data.

# A marginal risk difference equal to a threshold belongs to the null.
eps_comparison <- 1e-10

scenarios <- list(
  complete_null = list(pE=c(.300,.300,.305,.310), pS=c(.10,.70,.75,.80)),
  partial_null_efficacy = list(pE=c(.300,.305,.350,.400), pS=c(.10,.70,.75,.80)),
  partial_null_safety = list(pE=c(.300,.300,.305,.310), pS=c(.10,.20,.50,.78)),
  single_dose_window = list(pE=c(.300,.305,.350,.400), pS=c(.10,.20,.50,.78)),
  threshold_boundary = list(pE=c(.300,.310,.310,.350), pS=c(.10,.20,.70,.78))
)

# Joint binary P(E=1, S=1) lies in the Frechet interval.
q11_value <- function(pE, pS, mode) {
  lo <- pmax(0, pE+pS-1)
  hi <- pmin(pE,pS)
  switch(mode, negative=lo, independent=pE*pS, positive=hi,
         stop('Unknown dependence mode'))
}

# Draw n_rep independent trials per dose. Each patient has paired E/S outcomes.
# Multinomial distribution is generated with sequential binomials (vectorised).
simulate_counts <- function(n, pE, pS, mode, R) {
  q11 <- q11_value(pE,pS,mode)
  q10 <- pE-q11
  q01 <- pS-q11
  a11 <- rbinom(R, n, q11)
  left1 <- n-a11
  prob10 <- if (1-q11>0) q10/(1-q11) else 0
  a10 <- rbinom(R, left1, min(1,max(0,prob10)))
  left2 <- left1-a10
  rem <- q01+1-pE-pS+q11
  prob01 <- if (rem>0) q01/rem else 0
  a01 <- rbinom(R, left2, min(1,max(0,prob01)))
  list(E=a11+a10, S=a11+a01)
}

# Vectorised Farrington-Manning signed score for fixed null difference t.
# Constrained MLE p1=p0+t is obtained by bisection of the concave
# constrained log-likelihood's derivative, with boundary handling.
fm_z <- function(x1,n1,x0,n0,t,iterations=48L) {
  R <- max(length(x1),length(x0),length(t))
  x1 <- rep_len(x1,R); x0 <- rep_len(x0,R); t <- rep_len(t,R)
  low <- pmax(0,-t)
  high <- pmin(1,1-t)
  if (any(low>high)) stop('Infeasible risk difference')
  a <- low; b <- high
  deriv <- function(p0) {
    p0s <- pmin(1-1e-12,pmax(1e-12,p0))
    p1s <- pmin(1-1e-12,pmax(1e-12,p0+t))
    (x1-n1*p1s)/(p1s*(1-p1s)) + (x0-n0*p0s)/(p0s*(1-p0s))
  }
```

```
  # log-likelihood is concave; derivative is non-increasing in p0.
  # root outside range => boundary maximizer.
  for (iter in seq_len(iterations)) {
    mid <- (a+b)/2
    d <- deriv(mid)
    move_right <- d > 0
    a[move_right] <- mid[move_right]
    b[!move_right] <- mid[!move_right]
  }
  p0tilde <- (a+b)/2
  p1tilde <- p0tilde+t
  vv <- p1tilde*(1-p1tilde)/n1 + p0tilde*(1-p0tilde)/n0
  diff <- x1/n1-x0/n0-t
  z <- diff/sqrt(pmax(vv, .Machine$double.eps))
  z[vv < 1e-14 & abs(diff)<1e-12] <- 0
  z[vv < 1e-14 & abs(diff)>=1e-12] <- sign(diff[vv < 1e-14 & abs(diff)>=1e-12])*Inf
  z
}

# Exact CP limits for one binomial probability, with one-sided tail eta.
cp_lower <- function(x,n,eta) {
  out <- numeric(length(x)); good <- x>0
  out[good] <- qbeta(eta,x[good],n-x[good]+1)
  out
}
cp_upper <- function(x,n,eta) {
  out <- rep(1,length(x)); good <- x<n
  out[good] <- qbeta(1-eta,x[good]+1,n-x[good])
  out
}

# Sequential dose scanning from the highest (efficacy) and lowest (safety).
# Input matrices are R x k logical.
select_boundaries <- function(passE,passS,k) {
  R <- nrow(passE)
  # MinED = k+1 indicates 'not established'; MaxSD=0 likewise.
  M <- rep.int(k+1L,R)
  B <- integer(R)
  still <- rep(TRUE,R)
  for (i in k:1) {
    yes <- still & passE[,i]
    M[yes] <- i
    still <- yes
  }
  still <- rep(TRUE,R)
  for (i in seq_len(k)) {
    yes <- still & passS[,i]
    B[yes] <- i
    still <- yes
  }
  list(M=M,B=B,window=(M<=k & B>=1 & M<=B))
}

# Wilson interval for a Monte Carlo proportion, to quantify simulation error.
mc_ci <- function(x,R,z=qnorm(.975)) {
  p <- x/R; z2 <- z*z
  mid <- (p+z2/(2*R))/(1+z2/R)
  half <- z*sqrt(p*(1-p)/R + z2/(4*R*R))/(1+z2/R)
  c(max(0,mid-half), min(1,mid+half))
}

aggregate_results <- list()
dose_results <- list()
pos <- 0L
k <- 3L
crit <- qnorm(1-alpha/(2*k))
eta <- alpha/(2*(k+1))  # eight one-sided CP probability limits
cat(sprintf('FM critical value = %.6f; CP per-limit eta = %.6f\n',crit,eta))

for (sc_name in names(scenarios)) {
  ps <- scenarios[[sc_name]]
  pE <- ps$pE; pS <- ps$pS
  if (any(diff(pE)<0) || any(diff(pS)<0)) stop('Nonmonotone input scenario')
  trueE <- (pE[-1]-pE[1]) > (delta+eps_comparison)
  trueS <- (pS[-1]-pS[1]) < (gamma-eps_comparison)
  Mtrue <- if(any(trueE)) which(trueE)[1] else k+1L
  Btrue <- if(any(trueS)) tail(which(trueS),1) else 0L
  trueWindow <- (Mtrue<=k && Btrue>=1 && Mtrue<=Btrue)

  for (n in sample_sizes) for (dep in dependence_modes) {
```

```
    # Each column: control then doses 1,2,3; each row: one simulated trial.
    xE <- matrix(0L,n_rep,k+1L)
    xS <- matrix(0L,n_rep,k+1L)
    for (group in seq_len(k+1L)) {
      dat <- simulate_counts(n,pE[group],pS[group],dep,n_rep)
      xE[,group] <- dat$E; xS[,group] <- dat$S
    }
    passE_fm <- passS_fm <- passE_cp <- passS_cp <- matrix(FALSE,n_rep,k)
    score_region_inside <- rep(TRUE,n_rep)
    cp_region_inside <- rep(TRUE,n_rep)

    UE0 <- cp_upper(xE[,1],n,eta)   # one efficacy control UPPER bound
    LS0 <- cp_lower(xS[,1],n,eta)   # one safety control LOWER bound
    cp_region_inside <- cp_region_inside & (pE[1]<=UE0 & LS0<=pS[1])

    for (j in seq_len(k)) {
      col <- j+1L
      zEff <- fm_z(xE[,col],n,xE[,1],n,delta)
      zSaf <- fm_z(xS[,col],n,xS[,1],n,gamma)
      passE_fm[,j] <- zEff > crit
      passS_fm[,j] <- -zSaf > crit
      # Coverage diagnostic for the approximate maximum directional SCORE REGION
      # at the TRUE parameter vector, not the projected region bounds.
      zEff_true <- fm_z(xE[,col],n,xE[,1],n,pE[col]-pE[1])
      zSaf_true <- fm_z(xS[,col],n,xS[,1],n,pS[col]-pS[1])
      score_region_inside <- score_region_inside & (zEff_true<=crit) & (-zSaf_true<=crit)

      LEi <- cp_lower(xE[,col],n,eta)
      USi <- cp_upper(xS[,col],n,eta)
      passE_cp[,j] <- (LEi-UE0)>delta
      passS_cp[,j] <- (USi-LS0)<gamma
      cp_region_inside <- cp_region_inside & (LEi<=pE[col] & pS[col]<=USi)
    }

    for (method in c('FM_approx','CP_exact_benchmark')) {
      e <- if(method=='FM_approx') passE_fm else passE_cp
      s <- if(method=='FM_approx') passS_fm else passS_cp
      # elementary FWER = any false efficacy/safety declaration, BEFORE stepwise restriction
      erroneous <- rowSums(sweep(e,2,!trueE,'&'))>0 |
                   rowSums(sweep(s,2,!trueS,'&'))>0
      selected <- select_boundaries(e,s,k)
      # The stepwise rule's false declarations are a subset of elementary errors.
      # If additional reporting of sequential FWER is desired, construct its mask:
      seqE <- seqS <- matrix(FALSE,n_rep,k)
      for(j in seq_len(k)) {
        seqE[,j] <- (selected$M<=j & selected$M<=k)
        seqS[,j] <- (selected$B>=j & selected$B>=1)
      }
      seq_errors <- rowSums(sweep(seqE,2,!trueE,'&'))>0 |
                    rowSums(sweep(seqS,2,!trueS,'&'))>0
      inside <- if(method=='FM_approx') score_region_inside else cp_region_inside
      f <- sum(erroneous); cf <- mc_ci(f,n_rep)
      pos <- pos+1L
      aggregate_results[[pos]] <- data.frame(
        scenario=sc_name,n=n,dependence=dep,method=method,reps=n_rep,
        true_MinED=if(Mtrue==k+1) 'none' else as.character(Mtrue),
        true_MaxSD=if(Btrue==0) 'none' else as.character(Btrue),
        true_window=trueWindow,
        FWER_all_elementary=f/n_rep,FWER_CI_low=cf[1],FWER_CI_high=cf[2],
        FWER_stepwise=mean(seq_errors),
        FWER_stepwise_CI_low=mc_ci(sum(seq_errors),n_rep)[1],
        FWER_stepwise_CI_high=mc_ci(sum(seq_errors),n_rep)[2],
        simultaneous_coverage_diagnostic=mean(inside),
        correct_MinED=mean(selected$M==Mtrue),
        correct_MaxSD=mean(selected$B==Btrue),
        correct_boundary_pair=mean(selected$M==Mtrue & selected$B==Btrue),
        correct_window_status=mean(selected$window==trueWindow),
        reported_nonempty_window=mean(selected$window),
        stringsAsFactors=FALSE
      )
      dose_results[[pos]] <- data.frame(
        scenario=sc_name,n=n,dependence=dep,method=method,
        dose=c(1,2,3),true_efficacy=trueE,true_safety=trueS,
        efficacy_declaration_rate=colMeans(e),
        safety_declaration_rate=colMeans(s),stringsAsFactors=FALSE
      )
    }
    cat(sprintf('Finished %-24s n=%3d dependence=%-11s\n',sc_name,n,dep))
  }
}
```

```
summary_table <- do.call(rbind,aggregate_results)
dose_table <- do.call(rbind,dose_results)
file_summary <- file.path(out_dir,'phase6_simulation_summary.csv')
file_dose <- file.path(out_dir,'phase6_dosewise_power.csv')
write.csv(summary_table,file_summary,row.names=FALSE)
write.csv(dose_table,file_dose,row.names=FALSE)

cat('\nSummary of empirical FWER across all configurations:\n')
print(aggregate(FWER_all_elementary~method,summary_table,
                function(v)c(min=min(v),max=max(v),mean=mean(v))),row.names=FALSE)
cat('\nFiles written:\n',normalizePath(file_summary,winslash='/'),
    '\n',normalizePath(file_dose,winslash='/'),'\n')
cat('\nNOTE: FM is a normal-calibrated approximation, not exact C_alpha(Z).\n')
```

Supplementary Script S2

Extended simulation: two higher-efficacy scenarios

Filename: phase6_enhanced_verified_extended.R

```
# VERIFIED EXTENSION: preserves the original trial-generation sequence,
# original random seed, FWER_all_elementary and FWER_stepwise definitions.
# Adds separate correct_MinED and correct_MaxSD probabilities and CIs.
# Run BOTH this and the enhanced file, then combine their outputs.
# PHASE 6 FOLLOW-UP: Enhanced efficacy power simulation for 3 ordered doses
# Base R only. Uses a COMMON PATIENT COHORT per dose in the simulations.
# This differs from the published Sakai endpoint-specific analysis populations.
# Method FM: one-sided Farrington-Manning restricted-variance score tests,
# with normal Bonferroni critical value z_(1-alpha/(2*k)).
# The score region is approximate, NOT exact nuisance-calibrated C_alpha(Z).
# Method CP: exact one-sided Clopper-Pearson binomial bounds on 2*(k+1)
# distinct marginal probabilities, Bonferroni-adjusted, then differences.
# CP provides finite-sample simultaneous coverage without endpoint independence.

set.seed(20261104)
alpha <- 0.05
delta <- 0.01
gamma <- 0.60
n_rep <- 10000L       # Change to 1000L for a fast test before a full run.
sample_sizes <- c(50L, 100L, 200L)
dependence_modes <- c('negative', 'independent', 'positive')
out_dir <- getwd()     # Files save to YOUR R working directory, not /mnt/data.

# A marginal risk difference equal to a threshold belongs to the null.
eps_comparison <- 1e-10

# Targeted follow-up: stronger efficacy responses, same toxicity probabilities.
# Control efficacy 0.30 and toxicity 0.10. Dose 2 is truly MinED and MaxSD
# under both scenarios: E2-E0 > .01, S2-S0 < .60, while S3-S0 > .60.
scenarios <- list(
  enhanced_efficacy = list(pE=c(.300,.305,.450,.600), pS=c(.10,.20,.50,.78)),
  strong_efficacy   = list(pE=c(.300,.305,.550,.700), pS=c(.10,.20,.50,.78))
)

# Joint binary P(E=1, S=1) lies in the Frechet interval.
q11_value <- function(pE, pS, mode) {
  lo <- pmax(0, pE+pS-1)
  hi <- pmin(pE,pS)
  switch(mode, negative=lo, independent=pE*pS, positive=hi,
         stop('Unknown dependence mode'))
}

# Draw n_rep independent trials per dose. Each patient has paired E/S outcomes.
# Multinomial distribution is generated with sequential binomials (vectorised).
simulate_counts <- function(n, pE, pS, mode, R) {
  q11 <- q11_value(pE,pS,mode)
  q10 <- pE-q11
  q01 <- pS-q11
  a11 <- rbinom(R, n, q11)
  left1 <- n-a11
  prob10 <- if (1-q11>0) q10/(1-q11) else 0
  a10 <- rbinom(R, left1, min(1,max(0,prob10)))
  left2 <- left1-a10
  rem <- q01+1-pE-pS+q11
  prob01 <- if (rem>0) q01/rem else 0
  a01 <- rbinom(R, left2, min(1,max(0,prob01)))
  list(E=a11+a10, S=a11+a01)
}

# Vectorised Farrington-Manning signed score for fixed null difference t.
# Constrained MLE p1=p0+t is obtained by bisection of the concave
# constrained log-likelihood's derivative, with boundary handling.
fm_z <- function(x1,n1,x0,n0,t,iterations=48L) {
  R <- max(length(x1),length(x0),length(t))
  x1 <- rep_len(x1,R); x0 <- rep_len(x0,R); t <- rep_len(t,R)
  low <- pmax(0,-t)
  high <- pmin(1,1-t)
  if (any(low>high)) stop('Infeasible risk difference')
  a <- low; b <- high
  deriv <- function(p0) {
    p0s <- pmin(1-1e-12,pmax(1e-12,p0))
    p1s <- pmin(1-1e-12,pmax(1e-12,p0+t))
    (x1-n1*p1s)/(p1s*(1-p1s)) + (x0-n0*p0s)/(p0s*(1-p0s))
  }
```

```
  # log-likelihood is concave; derivative is non-increasing in p0.
  # root outside range => boundary maximizer.
  for (iter in seq_len(iterations)) {
    mid <- (a+b)/2
    d <- deriv(mid)
    move_right <- d > 0
    a[move_right] <- mid[move_right]
    b[!move_right] <- mid[!move_right]
  }
  p0tilde <- (a+b)/2
  p1tilde <- p0tilde+t
  vv <- p1tilde*(1-p1tilde)/n1 + p0tilde*(1-p0tilde)/n0
  diff <- x1/n1-x0/n0-t
  z <- diff/sqrt(pmax(vv, .Machine$double.eps))
  z[vv < 1e-14 & abs(diff)<1e-12] <- 0
  z[vv < 1e-14 & abs(diff)>=1e-12] <- sign(diff[vv < 1e-14 & abs(diff)>=1e-12])*Inf
  z
}

# Exact CP limits for one binomial probability, with one-sided tail eta.
cp_lower <- function(x,n,eta) {
  out <- numeric(length(x)); good <- x>0
  out[good] <- qbeta(eta,x[good],n-x[good]+1)
  out
}
cp_upper <- function(x,n,eta) {
  out <- rep(1,length(x)); good <- x<n
  out[good] <- qbeta(1-eta,x[good]+1,n-x[good])
  out
}

# Sequential dose scanning from the highest (efficacy) and lowest (safety).
# Input matrices are R x k logical.
select_boundaries <- function(passE,passS,k) {
  R <- nrow(passE)
  # MinED = k+1 indicates 'not established'; MaxSD=0 likewise.
  M <- rep.int(k+1L,R)
  B <- integer(R)
  still <- rep(TRUE,R)
  for (i in k:1) {
    yes <- still & passE[,i]
    M[yes] <- i
    still <- yes
  }
  still <- rep(TRUE,R)
  for (i in seq_len(k)) {
    yes <- still & passS[,i]
    B[yes] <- i
    still <- yes
  }
  list(M=M,B=B,window=(M<=k & B>=1 & M<=B))
}

# Wilson interval for a Monte Carlo proportion, to quantify simulation error.
mc_ci <- function(x,R,z=qnorm(.975)) {
  p <- x/R; z2 <- z*z
  mid <- (p+z2/(2*R))/(1+z2/R)
  half <- z*sqrt(p*(1-p)/R + z2/(4*R*R))/(1+z2/R)
  c(max(0,mid-half), min(1,mid+half))
}

aggregate_results <- list()
dose_results <- list()
pos <- 0L
k <- 3L
crit <- qnorm(1-alpha/(2*k))
eta <- alpha/(2*(k+1))  # eight one-sided CP probability limits
cat(sprintf('FM critical value = %.6f; CP per-limit eta = %.6f\n',crit,eta))

for (sc_name in names(scenarios)) {
  ps <- scenarios[[sc_name]]
  pE <- ps$pE; pS <- ps$pS
  if (any(diff(pE)<0) || any(diff(pS)<0)) stop('Nonmonotone input scenario')
  trueE <- (pE[-1]-pE[1]) > (delta+eps_comparison)
  trueS <- (pS[-1]-pS[1]) < (gamma-eps_comparison)
  Mtrue <- if(any(trueE)) which(trueE)[1] else k+1L
  Btrue <- if(any(trueS)) tail(which(trueS),1) else 0L
  trueWindow <- (Mtrue<=k && Btrue>=1 && Mtrue<=Btrue)

  for (n in sample_sizes) for (dep in dependence_modes) {
```

```
    # Each column: control then doses 1,2,3; each row: one simulated trial.
    xE <- matrix(0L,n_rep,k+1L)
    xS <- matrix(0L,n_rep,k+1L)
    for (group in seq_len(k+1L)) {
      dat <- simulate_counts(n,pE[group],pS[group],dep,n_rep)
      xE[,group] <- dat$E; xS[,group] <- dat$S
    }
    passE_fm <- passS_fm <- passE_cp <- passS_cp <- matrix(FALSE,n_rep,k)
    score_region_inside <- rep(TRUE,n_rep)
    cp_region_inside <- rep(TRUE,n_rep)

    UE0 <- cp_upper(xE[,1],n,eta)   # one efficacy control UPPER bound
    LS0 <- cp_lower(xS[,1],n,eta)   # one safety control LOWER bound
    cp_region_inside <- cp_region_inside & (pE[1]<=UE0 & LS0<=pS[1])

    for (j in seq_len(k)) {
      col <- j+1L
      zEff <- fm_z(xE[,col],n,xE[,1],n,delta)
      zSaf <- fm_z(xS[,col],n,xS[,1],n,gamma)
      passE_fm[,j] <- zEff > crit
      passS_fm[,j] <- -zSaf > crit
      # Coverage diagnostic for the approximate maximum directional SCORE REGION
      # at the TRUE parameter vector, not the projected region bounds.
      zEff_true <- fm_z(xE[,col],n,xE[,1],n,pE[col]-pE[1])
      zSaf_true <- fm_z(xS[,col],n,xS[,1],n,pS[col]-pS[1])
      score_region_inside <- score_region_inside & (zEff_true<=crit) & (-zSaf_true<=crit)

      LEi <- cp_lower(xE[,col],n,eta)
      USi <- cp_upper(xS[,col],n,eta)
      passE_cp[,j] <- (LEi-UE0)>delta
      passS_cp[,j] <- (USi-LS0)<gamma
      cp_region_inside <- cp_region_inside & (LEi<=pE[col] & pS[col]<=USi)
    }

    for (method in c('FM_approx','CP_exact_benchmark')) {
      e <- if(method=='FM_approx') passE_fm else passE_cp
      s <- if(method=='FM_approx') passS_fm else passS_cp
      # elementary FWER = any false efficacy/safety declaration, BEFORE stepwise restriction
      erroneous <- rowSums(sweep(e,2,!trueE,'&'))>0 |
                   rowSums(sweep(s,2,!trueS,'&'))>0
      selected <- select_boundaries(e,s,k)
      # The stepwise rule's false declarations are a subset of elementary errors.
      # If additional reporting of sequential FWER is desired, construct its mask:
      seqE <- seqS <- matrix(FALSE,n_rep,k)
      for(j in seq_len(k)) {
        seqE[,j] <- (selected$M<=j & selected$M<=k)
        seqS[,j] <- (selected$B>=j & selected$B>=1)
      }
      seq_errors <- rowSums(sweep(seqE,2,!trueE,'&'))>0 |
                    rowSums(sweep(seqS,2,!trueS,'&'))>0
      inside <- if(method=='FM_approx') score_region_inside else cp_region_inside
      f <- sum(erroneous); cf <- mc_ci(f,n_rep)
      pos <- pos+1L
      aggregate_results[[pos]] <- data.frame(
        scenario=sc_name,n=n,dependence=dep,method=method,reps=n_rep,
        true_MinED=if(Mtrue==k+1) 'none' else as.character(Mtrue),
        true_MaxSD=if(Btrue==0) 'none' else as.character(Btrue),
        true_window=trueWindow,
        FWER_all_elementary=f/n_rep,FWER_CI_low=cf[1],FWER_CI_high=cf[2],
        FWER_stepwise=mean(seq_errors),
        FWER_stepwise_CI_low=mc_ci(sum(seq_errors),n_rep)[1],
        FWER_stepwise_CI_high=mc_ci(sum(seq_errors),n_rep)[2],
        simultaneous_coverage_diagnostic=mean(inside),
        correct_MinED=mean(selected$M==Mtrue),
        correct_MaxSD=mean(selected$B==Btrue),
        correct_boundary_pair=mean(selected$M==Mtrue & selected$B==Btrue),
        correct_window_status=mean(selected$window==trueWindow),
        reported_nonempty_window=mean(selected$window),
        stringsAsFactors=FALSE
      )
      dose_results[[pos]] <- data.frame(
        scenario=sc_name,n=n,dependence=dep,method=method,
        dose=c(1,2,3),true_efficacy=trueE,true_safety=trueS,
        efficacy_declaration_rate=colMeans(e),
        safety_declaration_rate=colMeans(s),stringsAsFactors=FALSE
      )
    }
    cat(sprintf('Finished %-24s n=%3d dependence=%-11s\n',sc_name,n,dep))
  }
}
```

```
summary_table <- do.call(rbind,aggregate_results)
dose_table <- do.call(rbind,dose_results)
file_summary <- file.path(out_dir,'phase6_enhanced_efficacy_summary.csv')
file_dose <- file.path(out_dir,'phase6_enhanced_efficacy_dosewise.csv')
write.csv(summary_table,file_summary,row.names=FALSE)
write.csv(dose_table,file_dose,row.names=FALSE)

cat('\nSummary of empirical FWER across all configurations:\n')
print(aggregate(FWER_all_elementary~method,summary_table,
                function(v)c(min=min(v),max=max(v),mean=mean(v))),row.names=FALSE)
cat('\nFiles written:\n',normalizePath(file_summary,winslash='/'),
    '\n',normalizePath(file_dose,winslash='/'),'\n')
cat('\nNOTE: FM is a normal-calibrated approximation, not exact C_alpha(Z).\n')

# Focused output at n=200 for easiest interpretation.
cat('\nEnhanced efficacy scenarios, n=200: correct boundary pairs and FWER\n')
print(summary_table[summary_table$n == 200,
                    c('scenario','dependence','method',
                      'FWER_all_elementary','simultaneous_coverage_diagnostic',
                      'correct_boundary_pair','reported_nonempty_window')],
      digits=4, row.names=FALSE)
```

Supplementary Script S3

Combining simulation summaries and calculating pooled boundary-identification estimates

Filename: combine_verified_simulations_FIXED.R

```
# Combine verified baseline and enhanced simulations for manuscript Table 5.
# REQUIRED: run phase6_simulation_verified_extended.R, then
# phase6_enhanced_verified_extended.R, then THIS SCRIPT.
# Use a fresh R session (Session > Restart R) and source each script.

base_file <- 'phase6_simulation_summary.csv'
enh_file  <- 'phase6_enhanced_efficacy_summary.csv'
required <- c('scenario','n','dependence','method','reps',
              'correct_MinED','correct_MaxSD','correct_boundary_pair',
              'FWER_all_elementary','FWER_stepwise')
for (f in c(base_file, enh_file)) {
  if (!file.exists(f)) stop('Missing file: ', f, '\nWorking directory: ', getwd(), call.=FALSE)
}
base_data <- read.csv(base_file, check.names=FALSE)
enh_data  <- read.csv(enh_file, check.names=FALSE)
for (nm in c('baseline','enhanced')) {
  dat <- if (nm=='baseline') base_data else enh_data
  missing <- setdiff(required,names(dat))
  if (length(missing)) {
    stop('The ',nm,' summary is OLD or incomplete. Missing: ',
         paste(missing,collapse=', '), '\nRe-run ',
         if (nm=='baseline') 'phase6_simulation_verified_extended.R' else 'phase6_enhanced_verified_extended.R',
         ' to overwrite that summary, then re-run this script.',call.=FALSE)
  }
}
# Keep exactly the columns common to the two verified outputs.
if (!setequal(names(base_data),names(enh_data))) {
  stop('Output columns differ despite both files having required fields.\n',
       'Only baseline: ',paste(setdiff(names(base_data),names(enh_data)),collapse=', '),
       '\nOnly enhanced: ',paste(setdiff(names(enh_data),names(base_data)),collapse=', '),call.=FALSE)
}
enh_data <- enh_data[, names(base_data), drop=FALSE]
combined <- rbind(base_data,enh_data)
if (nrow(combined)!=126L) stop('Expected 126 rows (7 scenarios x 3 sample sizes x 3 dependencies x 2 methods); found
',nrow(combined),call.=FALSE)
if (any(combined$reps!=10000L)) stop('Some configurations do not have 10,000 replicates.',call.=FALSE)
if (any(combined$correct_boundary_pair > pmin(combined$correct_MinED,combined$correct_MaxSD)+1e-12)) {
  stop('Joint accuracy exceeds marginal accuracy in at least one configuration.',call.=FALSE)
}
needed <- c('single_dose_window','enhanced_efficacy','strong_efficacy')
t5 <- combined[combined$scenario %in% needed,,drop=FALSE]
if (nrow(t5)!=54L) stop('Expected 54 rows in Table 5 subset; found ',nrow(t5),call.=FALSE)
wilson <- function(p,N) {
  z <- qnorm(.975); a <- 1 + z^2/N
  center <- (p + z^2/(2*N))/a
  hw <- z*sqrt(p*(1-p)/N + z^2/(4*N*N))/a
  c(lower=max(0,center-hw), upper=min(1,center+hw))
}
keys <- unique(t5[,c('scenario','n','method')])
rows <- lapply(seq_len(nrow(keys)),function(h) {
  key <- keys[h,,drop=FALSE]
  z <- t5[t5$scenario==key$scenario & t5$n==key$n & t5$method==key$method,,drop=FALSE]
  if(nrow(z)!=3L || !setequal(z$dependence,c('negative','independent','positive'))) {
    stop('Expected all three dependence levels for ',key$scenario,', n=',key$n,', ',key$method,call.=FALSE)
  }
  N <- sum(z$reps)
  pm <- weighted.mean(z$correct_MinED,z$reps)
  pb <- weighted.mean(z$correct_MaxSD,z$reps)
  pj <- weighted.mean(z$correct_boundary_pair,z$reps)
  cim <- wilson(pm,N); cib <- wilson(pb,N); cij <- wilson(pj,N)
  data.frame(scenario=key$scenario,n=key$n,method=key$method,N=N,
             P_correct_MinED=pm,MinED_CI_lower=cim[1],MinED_CI_upper=cim[2],
             P_correct_MaxSD=pb,MaxSD_CI_lower=cib[1],MaxSD_CI_upper=cib[2],
             P_correct_both=pj,Both_CI_lower=cij[1],Both_CI_upper=cij[2])
})
table5 <- do.call(rbind,rows)
table5 <- table5[order(match(table5$scenario,needed),table5$n,table5$method),,drop=FALSE]
if(nrow(table5)!=18L) stop('Expected 18 combined table rows; found ',nrow(table5),call.=FALSE)
# Output only after all checks have passed.
write.csv(combined,'verified_all_63_configurations.csv',row.names=FALSE)
write.csv(table5,'verified_table5_extension.csv',row.names=FALSE)
cat('\nSUCCESS: 126 configuration/method rows, 18 Table 5 rows.\n')
cat('Saved in ',normalizePath(getwd(),winslash='/'), '\n',sep='')
print(table5[,c('scenario','n','method','P_correct_MinED','P_correct_MaxSD','P_correct_both')],row.names=FALSE,digits=5)
```