# Simulations for "Multiplicity adjustments for the Dunnett procedure under heteroscedasticity"

6/5/2023

This document provides simulation studies in Section 5 of "Multiplicity adjustments for the Dunnett procedure under heteroscedasticity" by Tamhane and Xi.

## Simulation function

```
setwd(folder)
source("functions.R")
sim <- function(x, arg) {
  mu <- unlist(arg$mu[x])
  sigma <- unlist(arg$sigma[x])
  n <- unlist(arg$n[x])
  direct <- arg$direction[x]
  nrep <- arg$nrep[x]
  alpha <- arg$alpha[x]
  k <- length(mu)
  padj0 <- padj1 <- padj2 <- padj3 <- matrix(0, nrow = nrep, ncol = k - 1)
  time0 <- time1 <- time2 <- time3 <- rep(NA, nrep)
  for(i in 1:nrep) {
    data <- data.frame(trt = rep(1:k, n), mu = rep(mu, n), resp = NA)
    muhat <- mu
    s <- sigma
    for (j in 1:k) {
      temp <- rnorm(n = n[j], mean = mu[j], sd = sigma[j])
      data$resp[data$trt == j] <- temp
      muhat[j] <- mean(temp)
      s[j] <- sqrt(sum((temp - muhat[j])^2) / (n[j] - 1))
    }
    summary <- data.frame(x = muhat, s2 = s^2, n = n)
    t <- c(NA, mu[-1])
    nu <- c(NA, mu[-1])
    for (j in 2:k) {
      t[j] <- (muhat[j] - muhat[1]) / sqrt(s[j]^2 / n[j] + s[1]^2 / n[1])
      nu[j] <- (s[j]^2 / n[j] + s[1]^2 / n[1])^2 /
        (s[j]^4 / n[j]^2 / (n[j] - 1) + s[1]^4 / n[1]^2 / (n[1] - 1))
    }
    summary <- data.frame(summary, t = t, nu = nu)
    # Method 0
    time <- proc.time()
    padj0[i, ] <- homo_func(x_0 = summary$x[1], x = summary$x[-1],
                            s2_0 = summary$s2[1], s2 = summary$s2[-1],
```

```
                            n_0 = summary$n[1], n = summary$n[-1],
                            direction = direct)
  time0 <- c(proc.time() - time)[3]
  # Method 1
  time <- proc.time()
  padj1[i, ] <- ind_func(x_0 = summary$x[1], x = summary$x[-1],
                           s2_0 = summary$s2[1], s2 = summary$s2[-1],
                           n_0 = summary$n[1], n = summary$n[-1],
                           direction = direct)
  time1 <- c(proc.time() - time)[3]
  # Method 2
  time <- proc.time()
  padj2[i, ] <- PI_func(x_0 = summary$x[1], x = summary$x[-1],
                          s2_0 = summary$s2[1], s2 = summary$s2[-1],
                          n_0 = summary$n[1], n = summary$n[-1],
                          direction = direct)
  time2 <- c(proc.time() - time)[3]
  # Method 3
  m <- 1e5
  time <- proc.time()
  padj3[i, ] <- sim_based_func(x_0 = summary$x[1], x = summary$x[-1],
                                 s2_0 = summary$s2[1], s2 = summary$s2[-1],
                                 n_0 = summary$n[1], n = summary$n[-1],
                                 direction = direct, nsim = m)
  time3 <- c(proc.time() - time)[3]
}

m0 <- m1 <- m2 <- m3 <- 0
dec0 <- padj0 <= alpha
dec1 <- padj1 <= alpha
dec2 <- padj2 <= alpha
dec3 <- padj3 <= alpha
if (all(mu[-1] - mu[1] == 0)) {
  m0 <- mean(apply(dec0, 1, max))
  m1 <- mean(apply(dec1, 1, max))
  m2 <- mean(apply(dec2, 1, max))
  m3 <- mean(apply(dec3, 1, max))
} else if (all(mu[-1] - mu[1] > 0)) {
  m0 <- m1 <- m2 <- m3 <- NA
} else {
  temp <- which(mu[-1] - mu[1] == 0)
  if (length(temp) == 1) {
    m0 <- mean(dec0[, temp])
    m1 <- mean(dec1[, temp])
    m2 <- mean(dec2[, temp])
    m3 <- mean(dec3[, temp])
  } else {
    m0 <- mean(apply(dec0[, temp], 1, max))
    m1 <- mean(apply(dec1[, temp], 1, max))
    m2 <- mean(apply(dec2[, temp], 1, max))
    m3 <- mean(apply(dec3[, temp], 1, max))
  }
}
```

```r
  out <- matrix(c(mu, n, sigma^2, nrep, alpha, mean(time0), m0, mean(time1),
                  m1, mean(time2), m2, mean(time3), m3), nrow = 1)
  colnames(out) <- c(paste0("mu_", 0:(k - 1)),
                     paste0("n_", 0:(k - 1)),
                     paste0("sigma_", 0:(k - 1),".2"),
                     "nrep",
                     "alpha",
                     "time0", "m0_fwer",
                     "time1", "m1_fwer",
                     "time2", "m2_fwer",
                     "time3", "m3_fwer"
  )
  return(out)
}
```

## Simulation scenarios for two treatment groups and control

We used the nominal one-sided $\alpha = 0.025$ and the total sample size $N = 30$. The following seven scenarios were considered where scenarios 1, 3, 4 and 5 were also considered by Hasler and Hornthon (2008). For each scenario, 100,000 simulated data sets were generated for each method under the global null hypothesis of $\mu_0 = \mu_1 = \mu_2 = 0$. For Method 3, 100,000 replicates were generated for each simulation.

1. Balanced allocation with equal standard deviations:
   $n_0 = 10, n_1 = 10, n_2 = 10$ and $\sigma_0 = 30, \sigma_1 = 30, \sigma_2 = 30$
2. Balanced allocation with control having the largest standard deviation:
   $n_0 = 10, n_1 = 10, n_2 = 10$ and $\sigma_0 = 50, \sigma_1 = 10, \sigma_2 = 10$
3. Balanced allocation with one treatment having the largest standard deviation:
   $n_0 = 10, n_1 = 10, n_2 = 10$ and $\sigma_0 = 10, \sigma_1 = 10, \sigma_2 = 50$
4. Unbalanced allocation with control having the smallest sample size and standard deviation:
   $n_0 = 4, n_1 = 13, n_2 = 13$ and $\sigma_0 = 10, \sigma_1 = 10, \sigma_2 = 50$
5. Unbalanced allocation with one treatment having the smallest sample size and the largest standard deviation:
   $n_0 = 13, n_1 = 13, n_2 = 4$ and $\sigma_0 = 10, \sigma_1 = 10, \sigma_2 = 50$
6. Unbalanced allocation with control having the smallest sample size and the largest standard deviation:
   $n_0 = 4, n_1 = 13, n_2 = 13$ and $\sigma_0 = 50, \sigma_1 = 10, \sigma_2 = 10$
7. Unbalanced allocation with a treatment having the smallest sample size and the smallest standard deviation:
   $n_0 = 13, n_1 = 13, n_2 = 4$ and $\sigma_0 = 50, \sigma_1 = 10, \sigma_2 = 10$

```r
arg <- data.frame(mu = I(list(rep(0, 3), rep(0, 3), rep(0, 3), rep(0, 3),
                              rep(0, 3), rep(0, 3), rep(0, 3),
                              c(0, 20, 0), c(0, 20, 0), c(0, 20, 0), c(0, 20, 0),
                              c(0, 20, 0), c(0, 20, 0), c(0, 20, 0))),
                  n = I(list(rep(10, 3), rep(10, 3), rep(10, 3), c(4, 13, 13),
                             c(13, 13, 4), c(4, 13, 13),
                             c(13, 13, 4),
                             rep(10, 3), rep(10, 3), rep(10, 3), c(4, 13, 13),
                             c(13, 13, 4), c(4, 13, 13),
                             c(13, 13, 4))),
                  sigma = I(list(rep(30, 3), c(50, 10, 10), c(10, 10, 50), c(10, 10, 50),
                                 c(10, 10, 50), c(50, 10, 10), c(50, 10, 10),
                                 rep(30, 3), c(50, 10, 10), c(10, 10, 50), c(10, 10, 50),
```

```
                               c(10, 10, 50), c(50, 10, 10), c(50, 10, 10))),
                  direction = "lower", nrep = 1e5, alpha = 0.025)
n_arg <- nrow(arg)

# # Parallel
# library(future.apply)
# plan(cluster)
# seed <- 10000
# result <- future_lapply(1:n_arg, FUN = sim, future.seed = seed,
#                         future.packages = c("mvtnorm"), arg = arg, future.scheduling = n_arg)
# results <- as.data.frame(do.call(rbind, result))
```

### Tables 1 and 2

```
results <- read.csv("two_treatment_groups.csv")
data_tbl1 <- subset(results, mu_1 == 0, select = -c(mu_0, mu_1, mu_2,
                                                    n_0, n_1, n_2,
                                                    sigma_0.2, sigma_1.2, sigma_2.2,
                                                    nrep, alpha,
                                                    time0, time1, time2, time3))
row.names(data_tbl1) <- 1:nrow(data_tbl1)
t(data_tbl1) # Table 1
```

```
##                1       2       3       4       5       6       7
## m0_fwer 0.02507 0.04760 0.03967 0.00450 0.17313 0.19703 0.02195
## m1_fwer 0.02307 0.01552 0.02397 0.02613 0.02651 0.01713 0.01753
## m2_fwer 0.02516 0.02343 0.02507 0.02914 0.02715 0.02436 0.02509
## m3_fwer 0.02455 0.02315 0.02419 0.02646 0.02664 0.02501 0.02455
```

```
data_tbl2 <- subset(results, mu_1 == 20, select = -c(mu_0, mu_1, mu_2,
                                                     n_0, n_1, n_2,
                                                     sigma_0.2, sigma_1.2, sigma_2.2,
                                                     nrep, alpha,
                                                     time0, time1, time2, time3))
row.names(data_tbl2) <- 1:nrow(data_tbl2)
t(data_tbl2) # Table 2
```

```
##                1       2       3       4       5       6       7
## m0_fwer 0.01343 0.03861 0.03874 0.00502 0.17146 0.18150 0.00414
## m1_fwer 0.01190 0.01292 0.01445 0.00828 0.02479 0.01531 0.01243
## m2_fwer 0.01307 0.01943 0.01310 0.01314 0.01554 0.02154 0.01834
## m3_fwer 0.01253 0.01927 0.01463 0.00851 0.02482 0.02242 0.01795
```

```
data_time <- subset(results, select = -c(mu_0, mu_1, mu_2,
                                         n_0, n_1, n_2,
                                         sigma_0.2, sigma_1.2, sigma_2.2,
                                         nrep, alpha,
                                         m0_fwer, m1_fwer, m2_fwer, m3_fwer))
round(colMeans(data_time), 4) # Mean computing time for each replication
```

```
##  time0  time1  time2  time3
## 0.0004 0.0000 0.0004 0.2589
```

## Simulation scenarios for three treatment groups and control

We used the nominal one-sided $\alpha = 0.025$ and the total sample size $N = 40$. The following seven scenarios were considered. For each scenario, 100,000 simulated data sets were generated for each method under the global null hypothesis of $\mu_0 = \mu_1 = \mu_2 = \mu_3 = 0$, under $\mu_0 = \mu_2 = \mu_3 = 0, \mu_1 = 20$, and under $\mu_0 = \mu_3 = 0, \mu_1 = \mu_2 = 20$. For Method 3, 100,000 replicates were generated for each simulation.

1. Balanced allocation with equal standard deviations: $n_0 = 10, n_1 = 10, n_2 = 10, n_3 = 10$ and $\sigma_0 = 30, \sigma_1 = 30, \sigma_2 = 30, \sigma_3 = 30$
2. Balanced allocation with control having the largest standard deviation: $n_0 = 10, n_1 = 10, n_2 = 10, n_3 = 10$ and $\sigma_0 = 50, \sigma_1 = 10, \sigma_2 = 10, \sigma_3 = 10$
3. Balanced allocation with one treatment having the largest standard deviation: $n_0 = 10, n_1 = 10, n_2 = 10, n_3 = 10$ and $\sigma_0 = 10, \sigma_1 = 10, \sigma_2 = 10, \sigma_3 = 50$
4. Unbalanced allocation with control having the smallest sample size and standard deviation: $n_0 = 4, n_1 = 12, n_2 = 12, n_3 = 12$ and $\sigma_0 = 10, \sigma_1 = 10, \sigma_2 = 10, \sigma_3 = 50$
5. Unbalanced allocation with one treatment having the smallest sample size and the largest standard deviation: $n_0 = 12, n_1 = 12, n_2 = 12, n_3 = 4$ and $\sigma_0 = 10, \sigma_1 = 10, \sigma_2 = 10, \sigma_3 = 50$
6. Unbalanced allocation with control having the smallest sample size and the largest standard deviation: $n_0 = 4, n_1 = 12, n_2 = 12, n_3 = 12$ and $\sigma_0 = 50, \sigma_1 = 10, \sigma_2 = 10, \sigma_3 = 10$
7. Unbalanced allocation with a treatment having the smallest sample size and the smallest standard deviation: $n_0 = 12, n_1 = 12, n_2 = 12, n_3 = 4$ and $\sigma_0 = 50, \sigma_1 = 10, \sigma_2 = 10, \sigma_3 = 10$

```r
arg <- data.frame(mu = I(list(rep(0, 4), rep(0, 4), rep(0, 4), rep(0, 4),
                              rep(0, 4), rep(0, 4), rep(0, 4),
                              c(0, 20, 0, 0), c(0, 20, 0, 0), c(0, 20, 0, 0),
                              c(0, 20, 0, 0), c(0, 20, 0, 0), c(0, 20, 0, 0),
                              c(0, 20, 0, 0),
                              c(0, 20, 20, 0), c(0, 20, 20, 0), c(0, 20, 20, 0),
                              c(0, 20, 20, 0), c(0, 20, 20, 0), c(0, 20, 20, 0),
                              c(0, 20, 20, 0))),
                  n = I(list(rep(10, 4), rep(10, 4), rep(10, 4), c(4, 12, 12, 12),
                             c(12, 12, 12, 4), c(4, 12, 12, 12),
                             c(12, 12, 12, 4),
                             rep(10, 4), rep(10, 4), rep(10, 4), c(4, 12, 12, 12),
                             c(12, 12, 12, 4), c(4, 12, 12, 12),
                             c(12, 12, 12, 4),
                             rep(10, 4), rep(10, 4), rep(10, 4), c(4, 12, 12, 12),
                             c(12, 12, 12, 4), c(4, 12, 12, 12),
                             c(12, 12, 12, 4))),
                  sigma = I(list(rep(30, 4), c(50, 10, 10, 10), c(10, 10, 10, 50),
                                 c(10, 10, 10, 50), c(10, 10, 10, 50),
                                 c(50, 10, 10, 10), c(50, 10, 10, 10),
                               rep(30, 4), c(50, 10, 10, 10), c(10, 10, 10, 50),
                                 c(10, 10, 10, 50), c(10, 10, 10, 50),
                               c(50, 10, 10, 10), c(50, 10, 10, 10),
                               rep(30, 4), c(50, 10, 10, 10), c(10, 10, 10, 50),
                                 c(10, 10, 10, 50), c(10, 10, 10, 50),
                               c(50, 10, 10, 10), c(50, 10, 10, 10))),
                  direction = "lower", nrep = 1e5, alpha = 0.025)
n_arg <- nrow(arg)

# # Parallel
# library(future.apply)
# plan(cluster)
```

```r
# seed <- 10000
# result <- future_lapply(1:n_arg, FUN = sim, future.seed = seed,
#                          future.packages = c("mvtnorm"), arg = arg,
# future.scheduling = n_arg)
# results <- as.data.frame(do.call(rbind, result))
```

#### Results for three treatment groups and control

```r
results <- read.csv("three_treatment_groups.csv")
# $\mu_0=\mu_1=\mu_2=\mu_3=0$
data_0_0_0_0 <- subset(results, mu_1 == 0 & mu_2 == 0,
                       select = -c(mu_0, mu_1, mu_2, mu_3,
                                   n_0, n_1, n_2, n_3,
                                   sigma_0.2, sigma_1.2, sigma_2.2, sigma_3.2,
                                   nrep, alpha,
                                   time0, time1, time2, time3))
row.names(data_0_0_0_0) <- 1:nrow(data_0_0_0_0)
t(data_0_0_0_0) # $\mu_0=\mu_1=\mu_2=\mu_3=0$
```

```
##                1       2       3       4       5       6       7
## m0_fwer 0.02501 0.06547 0.04882 0.01063 0.18052 0.21530 0.04250
## m1_fwer 0.02104 0.01234 0.02325 0.02390 0.02636 0.01301 0.01305
## m2_fwer 0.02441 0.02410 0.02573 0.02880 0.02800 0.02323 0.02426
## m3_fwer 0.02347 0.02371 0.02427 0.02596 0.02660 0.02341 0.02369
```

```r
# $\mu_0=\mu_2=\mu_3=0, \mu_1=20$
data_0_20_0_0 <- subset(results, mu_1 == 20 & mu_2 == 0,
                        select = -c(mu_0, mu_1, mu_2, mu_3,
                                    n_0, n_1, n_2, n_3,
                                    sigma_0.2, sigma_1.2, sigma_2.2, sigma_3.2,
                                    nrep, alpha,
                                    time0, time1, time2, time3))
row.names(data_0_20_0_0) <- 1:nrow(data_0_20_0_0)
t(data_0_20_0_0) # $\mu_0=\mu_2=\mu_3=0, \mu_1=20$
```

```
##                1       2       3       4       5       6       7
## m0_fwer 0.01820 0.06043 0.04899 0.01045 0.18000 0.20622 0.03502
## m1_fwer 0.01494 0.01057 0.01761 0.01698 0.02452 0.01203 0.01184
## m2_fwer 0.01769 0.02086 0.01825 0.02195 0.02091 0.02157 0.02172
## m3_fwer 0.01670 0.02053 0.01828 0.01866 0.02471 0.02153 0.02105
```

```r
# $\mu_0=\mu_3=0, \mu_1=\mu_2=20$
data_0_20_20_0 <- subset(results, mu_1 == 20 & mu_2 == 20,
                         select = -c(mu_0, mu_1, mu_2, mu_3,
                                    n_0, n_1, n_2, n_3,
                                    sigma_0.2, sigma_1.2, sigma_2.2, sigma_3.2,
                                    nrep, alpha,
                                    time0, time1, time2, time3))
row.names(data_0_20_20_0) <- 1:nrow(data_0_20_20_0)
t(data_0_20_20_0) # $\mu_0=\mu_3=0, \mu_1=\mu_2=20$
```

```
##                1       2       3       4       5       6       7
## m0_fwer 0.00975 0.04984 0.04880 0.01100 0.18125 0.18837 0.00957
## m1_fwer 0.00760 0.00903 0.01078 0.00472 0.02261 0.01092 0.00871
## m2_fwer 0.00911 0.01765 0.00964 0.00962 0.01197 0.01866 0.01589
## m3_fwer 0.00860 0.01732 0.01116 0.00546 0.02285 0.01886 0.01536
```

```
data_time <- subset(results, select = -c(mu_0, mu_1, mu_2, mu_3,
                                         n_0, n_1, n_2, n_3,
                                         sigma_0.2, sigma_1.2, sigma_2.2, sigma_3.2,
                                         nrep, alpha,
                                         m0_fwer, m1_fwer, m2_fwer, m3_fwer))
round(colMeans(data_time), 4) # Mean computing time for each replication
```

```
##  time0  time1  time2  time3
## 0.0040 0.0000 0.0044 0.4576
```

### Simulation scenarios for four treatment groups and control

We used the nominal one-sided $\alpha = 0.025$ and the total sample size $N = 50$. The following seven scenarios were considered. For each scenario, 100,000 simulated data sets were generated for each method under the global null hypothesis of $\mu_0 = \mu_1 = \mu_2 = \mu_3 = \mu_4 = 0$, under $\mu_0 = \mu_2 = \mu_3 = \mu_4 = 0, \mu_1 = 20$, under $\mu_0 = \mu_3 = \mu_4 = 0, \mu_1 = \mu_2 = 20$, and under under $\mu_0 = \mu_4 = 0, \mu_1 = \mu_2 = \mu_3 = 20$. For Method 3, 100,000 replicates were generated for each simulation.

1. Balanced allocation with equal standard deviations: $n_0 = 10, n_1 = 10, n_2 = 10, n_3 = 10, n_4 = 10$ and $\sigma_0 = 30, \sigma_1 = 30, \sigma_2 = 30, \sigma_3 = 30, \sigma_4 = 30$
2. Balanced allocation with control having the largest standard deviation: $n_0 = 10, n_1 = 10, n_2 = 10, n_3 = 10, n_4 = 10$ and $\sigma_0 = 50, \sigma_1 = 10, \sigma_2 = 10, \sigma_3 = 10, \sigma_4 = 10$
3. Balanced allocation with one treatment having the largest standard deviation: $n_0 = 10, n_1 = 10, n_2 = 10, n_3 = 10, n_4 = 10$ and $\sigma_0 = 10, \sigma_1 = 10, \sigma_2 = 10, \sigma_3 = 10, \sigma_4 = 50$
4. Unbalanced allocation with control having the smallest sample size and standard deviation: $n_0 = 6, n_1 = 11, n_2 = 11, n_3 = 11, n_4 = 11$ and $\sigma_0 = 10, \sigma_1 = 10, \sigma_2 = 10, \sigma_3 = 10, \sigma_4 = 50$
5. Unbalanced allocation with one treatment having the smallest sample size and the largest standard deviation: $n_0 = 11, n_1 = 11, n_2 = 11, n_3 = 11, n_4 = 6$ and $\sigma_0 = 10, \sigma_1 = 10, \sigma_2 = 10, \sigma_3 = 10, \sigma_4 = 50$
6. Unbalanced allocation with control having the smallest sample size and the largest standard deviation: $n_0 = 6, n_1 = 11, n_2 = 11, n_3 = 11, n_4 = 11$ and $\sigma_0 = 50, \sigma_1 = 10, \sigma_2 = 10, \sigma_3 = 10, \sigma_4 = 10$
7. Unbalanced allocation with a treatment having the smallest sample size and the smallest standard deviation: $n_0 = 11, n_1 = 11, n_2 = 11, n_3 = 11, n_4 = 6$ and $\sigma_0 = 50, \sigma_1 = 10, \sigma_2 = 10, \sigma_3 = 10, \sigma_4 = 10$

```
arg <- data.frame(mu = I(list(rep(0, 5), rep(0, 5), rep(0, 5), rep(0, 5),
                             rep(0, 5), rep(0, 5), rep(0, 5),
                             c(0, 20, 0, 0, 0), c(0, 20, 0, 0, 0), c(0, 20, 0, 0, 0),
                             c(0, 20, 0, 0, 0), c(0, 20, 0, 0, 0), c(0, 20, 0, 0, 0),
                             c(0, 20, 0, 0, 0),
                             c(0, 20, 20, 0, 0), c(0, 20, 20, 0, 0), c(0, 20, 20, 0, 0),
                             c(0, 20, 20, 0, 0), c(0, 20, 20, 0, 0), c(0, 20, 20, 0, 0),
                             c(0, 20, 20, 0, 0),
                             c(0, 20, 20, 20, 0), c(0, 20, 20, 20, 0), c(0, 20, 20, 20, 0),
                             c(0, 20, 20, 20, 0), c(0, 20, 20, 20, 0), c(0, 20, 20, 20, 0),
                             c(0, 20, 20, 20, 0))),
                  n = I(list(rep(10, 5), rep(10, 5), rep(10, 5), c(6, 11, 11, 11, 11),
                            c(11, 11, 11, 11, 6), c(6, 11, 11, 11, 11),
```

```
                             c(11, 11, 11, 11, 6),
                             rep(10, 5), rep(10, 5), rep(10, 5), c(6, 11, 11, 11, 11),
                             c(11, 11, 11, 11, 6), c(6, 11, 11, 11, 11),
                             c(11, 11, 11, 11, 6),
                             rep(10, 5), rep(10, 5), rep(10, 5), c(6, 11, 11, 11, 11),
                             c(11, 11, 11, 11, 6), c(6, 11, 11, 11, 11),
                             c(11, 11, 11, 11, 6),
                             rep(10, 5), rep(10, 5), rep(10, 5), c(6, 11, 11, 11, 11),
                             c(11, 11, 11, 11, 6), c(6, 11, 11, 11, 11),
                             c(11, 11, 11, 11, 6))),
                 sigma = I(list(rep(30, 5), c(50, 10, 10, 10, 10), c(10, 10, 10, 10, 50),
                                 c(10, 10, 10, 10, 50), c(10, 10, 10, 10, 50), c(50, 10, 10, 10, 10),
                                 c(50, 10, 10, 10, 10),
                                 rep(30, 5), c(50, 10, 10, 10, 10), c(10, 10, 10, 10, 50),
                                 c(10, 10, 10, 10, 50), c(10, 10, 10, 10, 50), c(50, 10, 10, 10, 10),
                                 c(50, 10, 10, 10, 10),
                                 rep(30, 5), c(50, 10, 10, 10, 10), c(10, 10, 10, 10, 50),
                                 c(10, 10, 10, 10, 50), c(10, 10, 10, 10, 50), c(50, 10, 10, 10, 10),
                                 c(50, 10, 10, 10, 10),
                                 rep(30, 5), c(50, 10, 10, 10, 10), c(10, 10, 10, 10, 50),
                                 c(10, 10, 10, 10, 50), c(10, 10, 10, 10, 50), c(50, 10, 10, 10, 10),
                                 c(50, 10, 10, 10, 10))),
                 direction = "lower", nrep = 1e5, alpha = 0.025)
n_arg <- nrow(arg)

# # Parallel
# library(future.apply)
# # plan(cluster)
# seed <- 10000
# result <- future_lapply(1:n_arg, FUN = sim, future.seed = seed,
#                          future.packages = c("mvtnorm"), arg = arg,
# future.scheduling = n_arg)
# results <- as.data.frame(do.call(rbind, result))
```

**Results for four treatment groups and control**

```
# $\mu_0=\mu_1=\mu_2=\mu_3=\mu_4=0$
results <- read.csv("four_treatment_groups.csv")
data_0_0_0_0_0 <- subset(results, mu_1 == 0 & mu_2 == 0 & mu_3 == 0,
                         select = -c(mu_0, mu_1, mu_2, mu_3, mu_4,
                                     n_0, n_1, n_2, n_3, n_4,
                                     sigma_0.2, sigma_1.2, sigma_2.2, sigma_3.2,
                                     sigma_4.2,
                                     nrep, alpha, time0, time1, time2, time3))
row.names(data_0_0_0_0_0) <- 1:nrow(data_0_0_0_0_0)
t(data_0_0_0_0_0) # $\mu_0=\mu_1=\mu_2=\mu_3=\mu_4=0$
```

```
##                1       2       3       4       5       6       7
## m0_fwer 0.02499 0.08130 0.05851 0.03171 0.13234 0.17086 0.06767
## m1_fwer 0.02015 0.01025 0.02243 0.02073 0.02331 0.00954 0.01067
## m2_fwer 0.02455 0.02330 0.02574 0.02516 0.02644 0.02270 0.02391
```

```
## m3_fwer 0.02286 0.02288 0.02402 0.02316 0.02445 0.02234 0.02346
```

```r
# $\mu_0=\mu_2=\mu_3=\mu_4=0, \mu_1=20$
data_0_20_0_0_0 <- subset(results, mu_1 == 20 & mu_2 == 0 & mu_3 == 0,
                          select = -c(mu_0, mu_1, mu_2, mu_3, mu_4,
                                      n_0, n_1, n_2, n_3, n_4,
                                      sigma_0.2, sigma_1.2, sigma_2.2, sigma_3.2,
                                      sigma_4.2,
                                      nrep, alpha, time0, time1, time2, time3))
row.names(data_0_20_0_0_0) <- 1:nrow(data_0_20_0_0_0)
t(data_0_20_0_0_0) # $\mu_0=\mu_2=\mu_3=\mu_4=0, \mu_1=20$
```

```
##               1       2       3       4       5       6       7
## m0_fwer 0.02021 0.07793 0.05803 0.03008 0.13038 0.16214 0.06174
## m1_fwer 0.01673 0.00992 0.01866 0.01645 0.02029 0.00852 0.00939
## m2_fwer 0.02007 0.02288 0.02066 0.02031 0.02166 0.02053 0.02219
## m3_fwer 0.01877 0.02251 0.02011 0.01849 0.02103 0.02022 0.02176
```

```r
# $\mu_0=\mu_3=\mu_4=0, \mu_1=\mu_2=20$
data_0_20_20_0_0 <- subset(results, mu_1 == 20 & mu_2 == 20 & mu_3 == 0,
                           select = -c(mu_0, mu_1, mu_2, mu_3, mu_4,
                                      n_0, n_1, n_2, n_3, n_4,
                                      sigma_0.2, sigma_1.2, sigma_2.2, sigma_3.2,
                                      sigma_4.2,
                                      nrep, alpha, time0, time1, time2, time3))
row.names(data_0_20_20_0_0) <- 1:nrow(data_0_20_20_0_0)
t(data_0_20_20_0_0) # $\mu_0=\mu_3=\mu_4=0, \mu_1=\mu_2=20$
```

```
##               1       2       3       4       5       6       7
## m0_fwer 0.01353 0.06941 0.05786 0.03116 0.13092 0.15241 0.05335
## m1_fwer 0.01083 0.00874 0.01374 0.01222 0.01730 0.00832 0.00875
## m2_fwer 0.01333 0.01990 0.01435 0.01538 0.01497 0.01942 0.01993
## m3_fwer 0.01234 0.01941 0.01462 0.01386 0.01805 0.01917 0.01938
```

```r
# $\mu_0=\mu_4=0, \mu_1=\mu_2=\mu_3=20$
data_0_20_20_20_0 <- subset(results, mu_1 == 20 & mu_2 == 20 & mu_3 == 20,
                            select = -c(mu_0, mu_1, mu_2, mu_3, mu_4,
                                      n_0, n_1, n_2, n_3, n_4,
                                      sigma_0.2, sigma_1.2, sigma_2.2, sigma_3.2,
                                      sigma_4.2,
                                      nrep, alpha, time0, time1, time2, time3))
row.names(data_0_20_20_20_0) <- 1:nrow(data_0_20_20_20_0)
t(data_0_20_20_20_0) # $\mu_0=\mu_4=0, \mu_1=\mu_2=\mu_3=20$
```

```
##               1       2       3       4       5       6       7
## m0_fwer 0.00808 0.05826 0.05769 0.03079 0.13132 0.13450 0.02930
## m1_fwer 0.00636 0.00664 0.00848 0.00575 0.01385 0.00780 0.00646
## m2_fwer 0.00773 0.01541 0.00770 0.00799 0.00870 0.01712 0.01530
## m3_fwer 0.00721 0.01504 0.00927 0.00686 0.01440 0.01688 0.01466
```

```r
data_time <- subset(results, select = -c(mu_0, mu_1, mu_2, mu_3, mu_4,
                                         n_0, n_1, n_2, n_3, n_4,
                                         sigma_0.2, sigma_1.2, sigma_2.2, sigma_3.2, sigma_4.2,
                                         nrep, alpha,
                                         m0_fwer, m1_fwer, m2_fwer, m3_fwer))
round(colMeans(data_time), 4) # Mean computing time for each replication
```

```
##  time0  time1  time2  time3
## 0.0090 0.0001 0.0090 0.6495
```

```r
sessionInfo()
```

```
## R version 4.1.2 (2021-11-01)
## Platform: x86_64-w64-mingw32/x64 (64-bit)
## Running under: Windows 10 x64 (build 19044)
##
## Matrix products: default
##
## locale:
## [1] LC_COLLATE=English_United States.1252
## [2] LC_CTYPE=English_United States.1252
## [3] LC_MONETARY=English_United States.1252
## [4] LC_NUMERIC=C
## [5] LC_TIME=English_United States.1252
##
## attached base packages:
## [1] stats     graphics  grDevices utils     datasets  methods   base
##
## loaded via a namespace (and not attached):
##  [1] compiler_4.1.2  magrittr_2.0.2  fastmap_1.1.0   cli_3.3.0
##  [5] tools_4.1.2     htmltools_0.5.2 rstudioapi_0.13 yaml_2.3.5
##  [9] stringi_1.7.6   rmarkdown_2.14  knitr_1.37      stringr_1.4.0
## [13] xfun_0.30       digest_0.6.29   rlang_1.0.2     evaluate_0.15
```