Sample size calculation for bioequivalence trials using pooled CV

Valery Liamtsau

2025-07-22

In this tutorial the calculaiton of the sample size will be based on the Cmax pharmacokinetic parameter

```
# reading data
library(readxl)
data <- read_excel("C:\\Users\\valer\\Desktop\\R_project\\pooledss.xlsx")

# data reprocessing
df_length <- nrow(data)          # length of dataframe
design <- data$Design            # seq of designs from the dataframe
n_subjects <- data$N_subjects    # number of subjects
CVs <- data$CV                   # seq of CVs

# printing pretty table
library(huxtable)
data_hux <-
  hux(data) |>
  set_bold(row = 1, col = everywhere, value = TRUE) |>
  set_all_borders(FALSE)
data_hux
```

The computation of the CV starts with the calculation of degrees of freedom (dfs) that are dependent upon the design of the bioequivalence study. The formulas for the calculation is presented below:

```
# to store the results after dfs calculation
dfs_vector <- vector(mode = 'list', length = df_length)
dfs_total <- 0
```

PKmetric	Study No.	Design	N_subjects	CV
Cmax	Study 1	2x2x2	44	30
Cmax	Study 2	2x2x2	47	28
Cmax	Study 3	2x2x2	35	35
Cmax	Study 4	2x2x4	22	37
Cmax	Study 5	2x2x4	24	25

```
dfs_total_seq <- NULL

# this function computes the degrees of freedom
dfs_calc <- function(df_length, design,
                     n_subjects, dfs_vector, k = 1, dfs_total,
                     dfs_total_seq) {

  # k is the pointer to iterate over all studies to computer the degrees
  # of freedom that depend on different design and number of subjects
  # iteration if performed until the end of the dataframe (df_length)

  while (k <= df_length) {

    if (design[k] == "2x2x2") {
      df = n_subjects[k] - 2
    } else if (design[k] == "3x3") {
      df = 2 * n_subjects[k] - 4
    } else if (design[k] == "3x6x3") {
      df = 2 * n_subjects[k] - 4
    } else if (design[k] == "4x4") {
      df = 3 * n_subjects[k] - 6
    } else if (design[k] == "2x2x3") {
      df = 2 * n_subjects[k] - 3
    } else if (design[k] == "2x2x4") {
      df = 3 * n_subjects[k] - 4
    } else if (design[k] == "2x4x4") {
      df = 3 * n_subjects[k] - 4
    } else if (design[k] == "2x3x2") {
      df = 3 * n_subjects[k] - 4
    }
```

```
    # store computed data
    dfs_total <- dfs_total + df
    dfs_vector[[k]] <- df
    dfs_total_seq <- c(dfs_total_seq, df)

    k <- k + 1
  }
  output <- list(dfs_vector, dfs_total, dfs_total_seq)
return (output)
}

output <- dfs_calc(df_length, design, n_subjects, dfs_vector,
                   k = 1, dfs_total, dfs_total_seq)
dfs_vector <- output[[1]]
dfs_total <- output[[2]]
dfs_total_seq <- output[[3]]
```

Degrees of freedom for all 5 studies (design dependent) listed in the input file are stored in the following sequence:

```
dfs_seq <- unlist(dfs_vector)
dfs_seq
```

```
[1] 42 45 33 62 68
```

Pooled CV computation

```
# this function computes pooled CV
CV_pooled_calc <- function(CVs, n_subjects) {

  # assigning different weights for each variance
  # the larger the number of subjects the more it impacts result
  # pooled CV = weighted sum (numerator) / total sum (denominator)

  numerator <- vector(mode = 'numeric', length = df_length)
  # computation of numerator of the pooled CV
  i <- 1
  while (i <= df_length) {
    numerator[i] <- (n_subjects[i] - 1) * CVs[i]
    i <- i + 1
  }
```

```
  # degrees of freedom calculation
  denumenator <- sum(n_subjects) - 2
  numerator <- sum(numerator)
  CVpooled <- numerator / denumenator

  return (CVpooled)
}

CVpooled <- CV_pooled_calc(CVs, n_subjects)
```

The following sequence will be pooled:

```
sprintf("The result of sequence pooling is %.2f", CVpooled)
```

```
[1] "The result of sequence pooling is 30.12"
```

The estimation of the sample size. Parameters definition

```
# this function makes equal group sizes
make_even_groups <- function(n) {
  return(as.integer(2 * (n %/% 2 + as.logical(n %% 2))))
}

CV         <- CVpooled/100    # total (pooled) CV from multiple studies
theta0     <- 0.95            # T/R-ratio (range 0.9 - 1.0)
theta1     <- 0.80            # lower BE-limit
theta2     <- 1.25            # upper BE-limit
target     <- 0.80            # desired (target) power
alpha      <- 0.05            # significance level of the test
beta       <- 1 - target      # type II error
df         <- data.frame(method = character(), # data frame to store values
                         iteration = integer(),
                         dfs = integer(),
                         CVpooled = integer(),
                         sample_size = integer(),
                         power = numeric())
s2         <- log(CV^2 + 1)   # conversion of CV to variance
s          <- sqrt(s2)        # standard deviation
z_alpha    <- qnorm(1 - alpha)
z_beta     <- qnorm(1 - beta)
z_beta_2   <- qnorm(1 - beta / 2)
```

Since the variance of the population is unknown, t approximation will be used for the intial sample size assessment

```
t_alpha <- qt(1 - alpha, dfs_total[[1]]) # different degrees of freedom

# central t approximation
if (theta0 == 1) {
  num   <- 2 * s2 * (z_beta_2 + t_alpha)^2
  number_groups <- ceiling(num / log(theta2)^2)
} else {
  num <- 2 * s2 * (z_beta + t_alpha)^2
  if (theta0 < 1) {
    denom <- (log(theta0) - log(theta1))^2
  } else {
    denom <- (log(theta0) - log(theta2))^2
  }
  number_groups <- ceiling(num / denom)
}

n <- make_even_groups(2 * number_groups)
t_alpha <- qt(1 - alpha, dfs_total[[1]])
power <- pnorm(sqrt((log(theta0) - log(theta2))^2 * number_groups/
                      (2 * s2)) - t_alpha) +
  pnorm(sqrt((log(theta0) - log(theta1))^2 * number_groups/
               (2 * s2)) - t_alpha) - 1
```

After the calculation is complete, the estimation using non central t approximation will be used for more precise sample size calculation

```
# applying noncentral t approximation
i <- 1
if (power < target) {# iterate upwards
  repeat {
    sem   <- sqrt(4 / n) * s
    ncp   <- c((log(theta0) - log(theta1)) / sem,
               (log(theta0) - log(theta2)) / sem)
    power <- diff(pt(c(+1, -1) * qt(1 - alpha,
                                    df = dfs_total),
                     df = dfs_total, ncp = ncp))
    df[i, 1:6] <- c("noncentral t", i, dfs_total,
                    sprintf("%.4f", CVpooled),n,
                    sprintf("%.4f", power))
```

```
    if (power >= target) {
      break
    }else {
      i <- i + 1
      n <- n + 2
    }
  }
} else {              # iterate downwards
  repeat {
    sem   <- sqrt(4 / n) * s # standard error of the mean
    # noncentrality parameter
    ncp   <- c((log(theta0) - log(theta1)) / sem,
    # degree to which mean of the test departs from the mean
    # when null hypostesis is true
               (log(theta0) - log(theta2)) / sem)
    power <- diff(pt(c(+1, -1) * qt(1 - alpha,
                                    df = dfs_total),
                     df = dfs_total, ncp = ncp))
    df[i, 1:6] <- c("noncentral t", i,dfs_total,
                    sprintf("%.4f", CVpooled), n,
                    sprintf("%.4f", power))
    if (power < target) {
      df <- df[-nrow(df), ]
      brea
      k
    }else {
      i <- i + 1
      n <- n - 2
    }
  }
}

df_hux_sample_size <-
  hux(df[nrow(df), ]) |>
  set_bold(row = 1, col = everywhere, value = TRUE) |>
  set_all_borders(FALSE)
df_hux_sample_size
```

method	iteration	dfs	CVpooled	sample_size	power
noncentral t	2	250	30.1176	76	0.8053

```
#print(df[nrow(df), ], row.names = FALSE)
```

The sample size calculation for other designs will be employed in case different clinical trials setups will be used

```
parallel <- as.integer(df$sample_size[-1])
crossover2x2x2 <- ceiling(parallel / 2)
crossover2x2x3 <- ceiling(parallel / 3)
crossover2x2x4 <- ceiling(parallel / 4)

sample_size <- c(parallel, crossover2x2x2, crossover2x2x3, crossover2x2x4)
designs <- c('parallel', '2x2x2', '2x2x3', '2x2x4')

df_samplesize         <- data.frame(designs, # data frame to store values
                                    sample_size)


df_hux_all_designs_sample_size <-
  hux(df_samplesize) |>
  set_bold(row = 1, col = everywhere, value = TRUE) |>
  set_all_borders(FALSE)
df_hux_all_designs_sample_size
```

designs	sample_size
parallel	76
2x2x2	38
2x2x3	26
2x2x4	19