Calculation of pharmacokinetic parameters for bioequivalence studies

Valery Liamtsau

2025-08-02

In this tutorial the calculation of pharmacokinetic parameters will be demonstrated using raw pharmacokinetic data

```
# upload packges
library(here) # to replace absolute path on relative
library(writexl)
library(pROC)
library(readxl)
library(tidyverse) # get for tibble
library(tidyverse)
library(lubridate)
library(hash)
library(zoo) # for AUC calculation
library(glue)
library(gghighlight)
library("psych") # for geom mean calculation
library(gt)
library(huxtable) # printing pretty table

# temporarily turn off warnings
options(warn=0)
```

Import and data preprocessing

```
# set working directory
setwd('/Users/valer/Desktop/R_project/')
path_ <- getwd()
path_ <- "C:/Users/valer/Desktop/R_project/project 3/data.xlsx"
```

```r
# loading data containing test and reference datasheets
loading_data <- function(path_, sheet_) {

  # reading file
  data <- read_excel(path = path_, sheet = sheet_)

  # convert all types to numeric to enable further calculations
  data <- data |>
    mutate_all(as.numeric)
  sum(is.na(data))
  return (data)

}

# dataset for the test medicinal product
data_test <- loading_data(path_, "test_rand")
```

Storing data

```r
# setting datastructues to store calculated data
# list of all r2 correlation coefficients
list_r2_test_product <- vector(mode = 'list', length = nrow(data_test))

# correlation coefficient
list_of_coeff_vs_r2_test_product <-
  vector(mode = 'list', length = ncol(data_test) - 1)

subject_list_data <- vector(mode = 'list', length = ncol(data_test))

# get the list of subjects to iterate over them later
subjects_list <- colnames(data_test)[2:ncol(data_test)]

# list of pk parameters
list_of_pharmacokinetic_parameters_test_product <-
  vector(mode = 'list', length = ncol(data_test) - 1)
```

Calculation of linear regression parameters. Looking for the best linear regression model

```r
for (subject in subjects_list) {

  data_work <- data_test[c('Time', subject)]
```

```
window_start <- nrow(data_work[subject])

# correction for the start
Cmax <- max(data_work[subject], na.rm = TRUE)

end <- which(data_work[[subject]] == Cmax) - 1 # need to substract one
# to include the Cmax value itselfin the assessment

# from start to iter start
while (is.na(data_work[window_start,subject]) == TRUE){
  window_start <- window_start - 1
}

window_end <- window_start - 2 # at least 3 points need to be used
# for the calculation of the linear model

# to grow until it is equal to end (Cmax)
r2max <- -1e5
lm_coefficient_range <- NULL
lm_r2_range <- NULL

# iterate over each row of small data frame
for (row in (1: nrow(data_work))) {

  # immediately end iteration because iteration is done
  # only to Cmax inclusive
  if (window_end == end) {
    break
  }

  # slice data frame to get y and x vectors for
  # linear regression: x - time, y - concentration
  # linear regression fit starts with min of 3 points
  # from the end of data frame
  # then iterate towards the beginning of data frame
  # getting increasing x and y by 1

  # start with 3 points
  # time range
  x <- data_work[["Time"]][window_end:window_start]

  # concentration range
```

```
y <- data_work[[subject]][window_end:window_start]
# need to log transform y coordinates (concentration)
y_log <- log(data_work[[subject]][window_end:window_start])

# fit a model, get coefficient and r2 values
model <- lm(formula = y_log ~ x,
            data = data_work[window_end:window_start,])
coefficient <- model$coefficients["x"]
r2 <- summary(model)$r.squared

# append coefficient to the coefficient range list
lm_coefficient_range <- c(lm_coefficient_range, coefficient)
# append r2 to the linear model r2 range list
lm_r2_range <- c(lm_r2_range, r2)

# use char as keys for list
r2 <- as.character(r2)
list_r2_test_product[[r2]] <- coefficient

# increase the number of points for the current window of x and y
# iteration from the end of data frame
window_end <- window_end - 1

# find the largest r2
if (r2 > r2max) {
  r2max <- max(r2max, r2) # correct this code later

  # select x and y values that showed maximal r2
  xvalues_bestfit <- x
  yvalues_bestfit <- y
}
elimination_constant_optimized <- list_r2_test_product[[r2max]]

output <- list(
  elimination_constant_optimized = elimination_constant_optimized,
  r2max = r2max,
  lm_coefficient_range = lm_coefficient_range,
  lm_r2_range = lm_r2_range,
  xvalues_bestfit = xvalues_bestfit,
  yvalues_bestfit = yvalues_bestfit,
  initial_df = data_work)
```

```
    subject_list_data[[subject]] <- output
  }
}
```

This function computes optimized r2 / coefficient and corresponding x/y values for the best fit of linear regression model.

```
# this function computes
pharmacokinetic_parameters_calculation <- function(data_temp) {

  # computing Cmax - max plasma concentration
  Cmax <- max(data_temp[subject])

  # computing the lowest concentration possible,
  # it usually at the longest time at the last row of data frame
  Clast <- data_temp[nrow(data_temp),][[subject]]

  # first timepoint (x coordinate) which showed the best fit
  # that resulted in the max eliminaiton constant
  Time_interval_start <- subject_list_data[[subject]]$xvalues_bestfit[1]

  # last timepoint (x coordinate) which showed the best fit
  # that resulted in the max eliminaiton constant
  Time_interval_end <- data_temp[nrow(data_temp),]["Time"]

  # Tmax - time at which Cmax was found
  Tmax <- which(data_temp[subject] == Cmax)

  # elimination constant was negative because of
  # the negative trent of the elimination model.
  # For furhter calculation it is needed to get absolute value
  elimination_constant_optimized <-
    abs(subject_list_data[[subject]]$elimination_constant_optimized)

  # elimination half-time, time at which 50 5 of drug is eliminated
  half_time <- round(log(2)/ elimination_constant_optimized, 2)

  # area under the curve (AUC) calculation
  # get coordinates for trapezoid
  x <- data_temp[["Time"]]
  y <- data_temp[[subject]]
  # order by index and get cumulative sume
```

```
  id <- order(x)
  AUC0_last <- sum(diff(x[id])*rollmean(y[id],2))

  # area under the curve at infinite time
  AUC0_inf <- AUC0_last + Clast/elimination_constant_optimized

  # residual area that is not covered (difference between
  # AUC0_inf and AUC0_last)
  residual_area <- (AUC0_inf-AUC0_last)/AUC0_inf * 100

  # store calculated values for each subject
  list_of_pharmacokinetic_parameters_test_product[[subject]] <-
    data.frame(Time_interval_start,
               Time_interval_end,
               elimination_constant_optimized,
               half_time, Clast, AUC0_last,
               AUC0_inf, residual_area, Cmax, Tmax)

  return (list_of_pharmacokinetic_parameters_test_product)
}
```

Iterate over all subjects and calculate corresponding pharmacokinetic data

```
# iterate over subjects
for (subject in subjects_list) {

  data_temp <- data_test |>
    select(1, subject) |>
    na.omit()

  list_of_pharmacokinetic_parameters_test_product <- pharmacokinetic_parameters_calculation(
}
```

```
# to create dataframe
pharmac_param_ci_calculation <-
  bind_rows(list_of_pharmacokinetic_parameters_test_product, .id = "No.")
pk_df <- as.data.frame(pharmac_param_ci_calculation)
```

```
# renaming columns in dataframe
new_names <- c('Sub.', 'T(s)', 'T(e)',
               'Kel', 'T1/2', 'C(l)', 'AUCt', 'AUCinf',
```

```
               'Area', 'Cmax', 'Tmax')
names(pk_df) <- new_names

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

Sub.	T(s)	T(e)	Kel	T1/2	C(l)	AUCt	AUCinf	Area	Cmax	Tmax
Subject_1	2	23	0.107	6.5	0.107	11	12	8.4	1.97	4
Subject_2	2.5	16	0.149	4.64	0.0995	6.41	7.08	9.42	1.23	2
Subject_3	8	16	0.153	4.54	0.0952	7.09	7.72	8.09	1.93	3
Subject_4	6	23	0.133	5.22	0.0535	8.39	8.79	4.58	1.9	3
Subject_5	2.5	16	0.142	4.87	0.113	7.1	7.9	10.1	1.28	5
Subject_6	12	23	0.109	6.37	0.0685	7.71	8.34	7.54	1.57	2
Subject_7	10	16	0.206	3.37	0.0556	6.25	6.52	4.15	2.21	3
Subject_8	10	16	0.2	3.47	0.0471	4.68	4.92	4.8	1.32	2
Subject_9	2.5	23	0.0986	7.03	0.0899	7.28	8.19	11.1	0.996	3
Subject_10	1.75	23	0.142	4.88	0.0503	7.7	8.05	4.39	1.63	4