R CODE

```
# cx_mo_to_rec_fol = time to event in days
# dfs = indicator for recurrence or death (whichever occurs first)
# mito = adjuvant mitotane treatment, Si = Yes, No = No
# ki67 = Ki-67 proliferation index (%)
# age = patient’s age in years # N = number of lymph nodes
# margen = surgical margin (r0 vs r1/2)
# symp_horm_hyper = hormonal hypersecretion syndrome
# sex = patient’s sex
# rad = radiotherapy treatment (Yes/No)
# size_PT = primary tumor size
# year = year of diagnosis

library("devtools")
library("mice")
library("mixcurelps") # devtools::install_github("oswaldogressani/mixcurelps")
library("survival")
library(readxl)
library(dplyr)
library(Matrix)

rm(list = ls())
specific_df2 <- read_excel("enter_your_path_here/mitotane.xlsx")

mnum <- 20  # Number of imputations
imp <- mice(specific_df2, m = mnum, maxit = 50, method = 'pmm', seed = 1234)
imputed_datasets <- complete(imp, action = "all")

dataimplist <- list()
for(j in 1:mnum) {
  data <- complete(imp, j)
  data <- data[-which(is.na(data$margen)), ]
  colnames(data) <- c("y", "dfs", "mito", "ki67", "age", "n", "margen", "symp", "sex", "rad", "size", "year")
  data$mito <- ifelse(data$mito == "Si", 1, 0)
  data$margen <- ifelse(data$margen == "r0", 1, 0)
  data$symp <- ifelse(data$symp == "Si", 1, 0)
  data$rad <- ifelse(data$rad == "Si", 1, 0)
  data$sex <- ifelse(data$sex == "Male", 1, 0)
  dataimplist[[j]] <- data
}

# Define and fit the model on each imputed dataset using the 'lpsmc' function
formula <- Surv(y, dfs) ~
  inci(mito + ki67 + age + n + margen + symp + sex + rad + size + year) +
  late(mito + ki67 + age + n + margen + size + symp +sex)

K <- 15
p <- 11
q <- 8
fit1 <- lpsmc(formula, data = dataimplist[[1]], penorder = 2, checkPD = FALSE,
              stepsize = 0.1, v0 = 12, K = K)
fit2 <- lpsmc(formula, data = dataimplist[[2]], penorder = 2, checkPD = FALSE,
              stepsize = 0.1, v0 = 12, K = K)
fit3 <- lpsmc(formula, data = dataimplist[[3]], penorder = 2, checkPD = FALSE,
              stepsize = 0.1, v0 = 16, K = K)
fit4 <- lpsmc(formula, data = dataimplist[[4]], penorder = 2, checkPD = FALSE,
              stepsize = 0.1, K = K)
fit5 <- lpsmc(formula, data = dataimplist[[5]], penorder = 2, checkPD = FALSE,
              stepsize = 0.2, v0 = 13, K = K)
fit6 <- lpsmc(formula, data = dataimplist[[6]], penorder = 2, checkPD = FALSE,
              stepsize = 0.2, v0 = 13, K = K)
fit7 <- lpsmc(formula, data = dataimplist[[7]], penorder = 2, checkPD = FALSE,
              stepsize = 0.2, v0 = 15, K = K)
fit8 <- lpsmc(formula, data = dataimplist[[8]], penorder = 2, checkPD = FALSE,
              stepsize = 0.2, v0 = 15, K = K)
```

```
fit9 <- lpsmc(formula, data = dataimplist[[9]], penorder = 2, checkPD = FALSE,
              stepsize = 0.2, v0 = 13, K = K)
fit10 <- lpsmc(formula, data = dataimplist[[10]], penorder = 2, checkPD = FALSE,
              stepsize = 0.2, v0 = 15, K = K)
fit11 <- lpsmc(formula, data = dataimplist[[11]], penorder = 2, checkPD = FALSE,
               stepsize = 0.2, v0 = 15, K = K)
fit12 <- lpsmc(formula, data = dataimplist[[12]], penorder = 2, checkPD = FALSE,
               stepsize = 0.2, v0 = 13, K = K)
fit13 <- lpsmc(formula, data = dataimplist[[13]], penorder = 2, checkPD = FALSE,
               stepsize = 0.2, v0 = 13, K = K)
fit14 <- lpsmc(formula, data = dataimplist[[14]], penorder = 2, checkPD = FALSE,
               stepsize = 0.2, v0 = 13, K = K)
fit15 <- lpsmc(formula, data = dataimplist[[15]], penorder = 2, checkPD = FALSE,
               stepsize = 0.2, v0 = 13, K = K)
fit16 <- lpsmc(formula, data = dataimplist[[16]], penorder = 2, checkPD = FALSE,
               stepsize = 0.2, v0 = 13, K = K)
fit17 <- lpsmc(formula, data = dataimplist[[17]], penorder = 2, checkPD = FALSE,
               stepsize = 0.1, v0 = 12, K = K)
fit18 <- lpsmc(formula, data = dataimplist[[18]], penorder = 2, checkPD = FALSE,
               stepsize = 0.1, v0 = 12, K = K)
fit19 <- lpsmc(formula, data = dataimplist[[19]], penorder = 2, checkPD = FALSE,
               stepsize = 0.2, v0 = 17, K = K)
fit20 <- lpsmc(formula, data = dataimplist[[20]], penorder = 2, checkPD = FALSE,
               stepsize = 0.1, v0 = 13, K = K)

# Extract coefficients (betahat and gammahat) and covariance matrices (Covhat) from each fitted model
estimates <- lapply(list(fit1, fit2, fit3, fit4, fit5,
                         fit6, fit7, fit8, fit9, fit10,
                         fit11, fit12, fit13, fit14, fit15,
                         fit16, fit17, fit18, fit19, fit20), function(fit) {
  list(betahat = fit$betahat, gammahat = fit$gammahat, cov = fit$Covhat)
})

Q_bar_beta <- Reduce("+", lapply(estimates, function(x) x$betahat)) / mnum
Q_bar_gamma <- Reduce("+", lapply(estimates, function(x) x$gammahat)) / mnum

B_beta <- Reduce("+", lapply(estimates, function(x) {
  (x$betahat - Q_bar_beta) %*% t(x$betahat - Q_bar_beta)
})) / (mnum - 1)

B_gamma <- Reduce("+", lapply(estimates, function(x) {
  (x$gammahat - Q_bar_gamma) %*% t(x$gammahat - Q_bar_gamma)
})) / (mnum - 1)

U <- Reduce("+", lapply(estimates, function(x) x$cov)) / mnum

U_combined <- bdiag(Matrix(U[(K+1):(K+p),(K+1):(K+p)], sparse = TRUE),
                    Matrix(U[(K+p+1):(K+p+q), (K+p+1):(K+p+q)], sparse = TRUE))
B_combined <- bdiag(Matrix(B_beta, sparse = TRUE), Matrix(B_gamma, sparse = TRUE))

if (all(dim(U_combined) == dim(B_combined))) {
  total_variance <- U_combined + (1 + (1 / mnum)) * B_combined
} else {
  stop("Dimensions of U_combined and B_combined do not match.")
}

std_errors <- sqrt(diag(total_variance))
Q_bar <- c(Q_bar_beta, Q_bar_gamma)  # Combine Q_bar_beta and Q_bar_gamma into a single vector
z_scores <- Q_bar / std_errors
p_values <- 2 * pnorm(abs(z_scores), lower.tail = FALSE)

ci_lower <- Q_bar - 1.96 * std_errors
ci_upper <- Q_bar + 1.96 * std_errors

variable_names <- c("Intercept", "mito", "ki67", "age", "n", "margen",
                    "symp", "sex", "rad", "size", "year",
                    "late_mito", "late_ki67", "late_age", "late_n",
                    "late_margen", "late_size", "late_symp", "late_sex")

options(scipen = 999)

final_model <- data.frame(
  Variable = variable_names,
  Estimate = round(Q_bar, 3),
  StdError = round(std_errors, 3),
  Z = round(z_scores, 3),
  PValue = round(p_values, 3),
  CI_Lower_95 = round(ci_lower, 3),
  CI_Upper_95 = round(ci_upper, 3),
  exp_est = round(exp(Q_bar), 3),
  exp_cil = round(exp(ci_lower), 3),
  exp_ciu = round(exp(ci_upper), 3)
)

final_model
```