Disease model for rhythm control strategies in symptomatic AF patients

Simone Huygens and Matthijs Versteegh on behalf of Zorginstituut Nederland

2025-05-07

For more information on the input parameters and methods, we refer to the technical report.

# 01 Settings and load required packages, functions and data

# 01.0 Clear memory

rm(list = ls()) # clear memory (removes all the variables from the workspace)  
options(scipen=999) # disable scientific notation  
set.seed(1) # set the seed for random number generation  
  
#### Adjustment for inflation using consumer price index ####  
# Cost year = 2024, except for drug costs (2025)  
# Correct for inflation  
adjust\_inflation <- function(amount, source\_year, target\_year = 2024) {  
 # Define yearly inflation rates (as decimals)  
 infl\_rates <- c(  
 "2017" = 0.014,  
 "2018" = 0.016,  
 "2019" = 0.012,  
 "2020" = 0.025,  
 "2021" = 0.118,  
 "2022" = 0.030,  
 "2023" = 0.024  
 )  
   
 # Get years to include in calculation  
 years <- as.character(source\_year:(target\_year-1))  
   
 # Calculate inflation factor and apply  
 factor <- prod(1 + infl\_rates[years])  
 return(amount \* factor)  
}

## 01.1 Load packages

## 01.2 Settings

# Model settings   
min\_age <- 18 # min age: only include adults  
max\_age <- 108 # max age based on max in Human Mortality Database  
cl <- 0.5 # cycle length in years: should be 0.5 because not all   
 # input parameters are automatically varied (i.e. TTE Vektis)  
n\_t <- (max\_age-min\_age)/cl # time horizon in cycles  
n\_cycles <- n\_t # number of cycles   
n\_i <- 50000 # number of simulated individuals (n\_i >= 50,000 for stable model results)  
v\_n <- c("SFAF", "SAF", "D") # model states names  
v\_names\_states <- v\_n # variable needed for plot\_trace\_microsim  
n\_states <- n\_s <- length(v\_n) # the number of health states  
wtp <- 20000 # willingness to pay threshold (based on proportional shortfall AF)  
d\_rc <- 0.030 # discount rate costs  
d\_re <- 0.015 # discount rate effects  
d\_rc <- (1+d\_rc)^cl-1 # discount rate costs to x-weekly   
d\_re <- (1+d\_re)^cl-1 # discount rate effects to x-weekly   
v\_dwc <- 1 / ((1 + d\_rc) ^ (0:n\_t)) # per period discount weight costs  
v\_dwe <- 1 / ((1 + d\_re) ^ (0:n\_t)) # per period discount weight effects  
v\_wcc <- darthtools::gen\_wcc(n\_cycles = n\_t, method = "Simpson1/3") # within-cycle correction (WCC)   
v\_names\_lines <- c("TRT1", "TRT2", "TRT3", "TRT4", "TRT5", "TRT6", "no-treatment", "D") # line names  
  
# Base case or scenario  
# The scenario "TTE\_2CV" includes the time to event data from the health insurers declaration data (Vektis) of the scenario analysis where AF symptoms are defined as re-ablation, His-ablation, MAZE procedure, switch to amiodarone or at least 2 cardioversions within one year (i.e. single cardioversions without a second cardioversion within a year are not counted as recurrence of AF symptoms). In the base case all cardioversions are counted as recurrence of AF symptoms after which a treatment switch will be incurred in the model.  
  
scenario <- "basecase" # "basecase" or "TTE\_2CV"

## 01.3 Load functions

# Load generic functions  
source(here::here("functions", "functions.R"))

# 02 Define parameter input values

# 02 Defineparameter input values  
#### PATIENT CHARACTERISTICS ####  
n\_survived <- 30182 # de Mol et al. 2022 (NHR 2013-2020) Table 1  
n\_deceased <- 15 # de Mol et al. 2022 (NHR 2013-2020) Table 1  
n\_total <- n\_survived+n\_deceased # de Mol et al. 2022 (NHR 2013-2020) Table 1   
age <- ((61.4\*n\_survived)+(65.9\*n\_deceased))/n\_total # de Mol et al. 2022 (NHR 2013-2020) Table 1   
sd\_age <- ((9.8\*n\_survived)+(7.1\*n\_deceased))/n\_total # de Mol et al. 2022 (NHR 2013-2020) Table 1   
n\_female <- 9816+7 # de Mol et al. 2022 (NHR 2013-2020) Table 1 (survived + deceased)  
n\_male <- n\_total-n\_female   
prop\_female <- n\_female/(n\_female+n\_male)  
n\_paroxysmal <- 19527+11 # proportion paroxysmal AF: de Mol et al. 2022 (NHR 2013-2020) Table 1 (survived + deceased)  
n\_persistent <- 7300+3+542+0 # proportion persistent AF: de Mol et al. 2022 (NHR 2013-2020) Table 1 (survived + deceased)  
prop\_parox <- n\_paroxysmal/(n\_paroxysmal+n\_persistent)  
  
# Generate patient population (will be varied in PSA)  
set.seed(2)  
v\_age <- round(rtruncnorm(n\_i, min\_age, max\_age, age, sd\_age))   
v\_Sex <- rbinom(n\_i, 1, prop\_female)   
  
#### CLINICAL INPUTS ####  
  
#### Probabilities of treatment success ####  
# Load results of meta-analysis Cochrane Nederland  
load(here::here("input", "meta-analysis-efficacy.RData"))  
load(here::here("input", "meta-analyse disc\_AAD.RData"))  
  
# Relative risks of AF recurrence AAD vs. CA (Source: meta-analysis Cochrane Nederland)   
RR\_nai <- exp(predrema1$pred) # 1.76 95% CI 1.34-2.32. Patients naive to AAD and CA, i.e. first line.   
RR\_AAD\_exp\_parox <- exp(predrema2$pred) # 2.32 95% CI 1.78-3.01. Patients with paroxysmal AF exposed to AAD but naive to CA  
RR\_AAD\_exp\_pers <- exp(predrema3$pred) # 1.62 95% CI 1.4-1.87. Patients with persistent AF exposed to AAD but naive to CA  
RR\_CA\_exp <- 2.12 # CI 1.61-2.80, based on 1 study. Patients exposed to CA and naive or exposed to AAD.  
  
# Probability of recurrence of AF after CA after 1 year   
prop\_AF\_CA\_nai <- plogis(predrema1i$pred) # 0.18 CI 0.08-0.35. After 1st CA in AAD/CA naive patients (Source: meta-analysis)  
prop\_AF\_CA\_AAD\_exp\_parox <- plogis(predrema2i$pred) # 0.26 CI 0.18-0.36. After 1st CA in AAD exposed paroxysmal AF patients (Source: meta-analysis)  
prop\_AF\_CA\_AAD\_exp\_pers <- plogis(predrema3i$pred) # 0.38 CI 0.33-0.42. After 1st CA in AAD exposed persistent AF patients. (Source: meta-analysis)  
prop\_AF\_CA\_CA\_exp\_1 <- 0.2579740 # After 2nd CA. (Source: Vektis)  
prop\_AF\_CA\_CA\_exp\_2 <- 0.3292506 # After 3+ CA. (Source: Vektis)  
  
#### Probability of recurrence of AF symptoms after CA over time ####  
# Load transition probabilities for recurrence of AF symptoms based on data from the health insurers declaration data (Vektis)  
if(scenario == "basecase"){  
 load(here::here("input", "TTE\_AF\_Vektis.RData")) # survival model objects (version 17-04-2025)  
   
 # Generate transition probabilities from survival model objects (requires flexsurv package 2.3.2)  
 df\_s\_ablation1 <- summary(best\_fit\_ablation1, t = seq(0, n\_t\*cl, cl), ci = T, newdata = T)  
 p\_SFAF\_CA\_SAF\_L1 <- trans\_prob(df\_s\_ablation1[[1]]$est) # probability of recurrence of AF after first CA  
 df\_s\_ablation2 <- summary(best\_fit\_ablation2, t = seq(0, n\_t\*cl, cl), ci = T, newdata = T)  
 p\_SFAF\_CA\_SAF\_L2 <- trans\_prob(df\_s\_ablation2[[1]]$est) # probability of recurrence of AF after second CA  
 df\_s\_ablation3 <- summary(best\_fit\_ablation3, t = seq(0, n\_t\*cl, cl), ci = T, newdata = T)  
 p\_SFAF\_CA\_SAF\_L3 <- trans\_prob(df\_s\_ablation3[[1]]$est)  
}  
  
if(scenario == "TTE\_2CV"){  
 load(here::here("input", "TTE\_AF\_Vektis\_scenario.RData")) # survival model objects (version 17-04-2025)  
   
 # Generate transition probabilities from survival model objects (requires flexsurv package 2.3.2)  
 df\_s\_ablation1 <- summary(best\_fit\_ablation1, t = seq(0, n\_t\*cl, cl), ci = T, newdata = T)  
 p\_SFAF\_CA\_SAF\_L1 <- trans\_prob(df\_s\_ablation1[[1]]$est) # probability of recurrence of AF after first CA  
 df\_s\_ablation2 <- summary(best\_fit\_ablation2, t = seq(0, n\_t\*cl, cl), ci = T, newdata = T)  
 p\_SFAF\_CA\_SAF\_L2 <- trans\_prob(df\_s\_ablation2[[1]]$est) # probability of recurrence of AF after second CA  
 p\_SFAF\_CA\_SAF\_L3 <- p\_SFAF\_CA\_SAF\_L2 # in scenario analysis the probability of recurrence of AF after is pooled for second or more CA  
}  
  
#### Probability of death ####  
# Background mortality in the general population weighted by the sex distribution of the patient population  
# We used 2019 to avoid bias from COVID-19  
df\_mort <- read.csv(here::here("input", "sterfte\_NL\_2019.csv"), sep = ";", header = TRUE)  
df\_mort <- df\_mort[ , -c(1,5)] # remove year and total column  
df\_mort <- df\_mort %>% # convert table from wide to long with a variable for sex  
 pivot\_longer(  
 cols = c(Female, Male),  
 names\_to = "Sex",  
 values\_to = "r\_mort"  
 ) %>%  
 mutate(  
 Sex = ifelse(Sex == "Female", 1, 0),  
 r\_mort = as.numeric(r\_mort)  
 )  
df\_mort$r\_mort\_cl <- df\_mort$r\_mort\*cl # convert annual mortality rate to x-weekly mortality rate  
df\_mort$p\_mort\_cl <- 1-exp(-df\_mort$r\_mort\_cl) # convert x-weekly mortality rate to probability  
df\_mort[df\_mort$Age == 108, "p\_mort\_cl"] <- 1 # set mortality rate to 1 at max age to make sure everyone dies in the model  
  
# Excess mortality in patients with AF based on Vinter et al. (2020)  
# HR of 'Model adjusted for baseline covariates' of 2001-15  
# We did not use 'Model adjusted for time varying covariates' because it was no proportional hazards model  
HR\_EM <- 2.01 # 95% CI: 1.71 to 2.36 (Vinter et al. 2020)  
  
#### Probability of severe adverse events ####  
# Discontinuation of AAD's due to adverse events  
p\_disc\_AAD\_nai <- 0.121 # Source: meta-analysis Cochrane Netherlands  
p\_disc\_AAD\_exp <- 0.225 # Source: meta-analysis Cochrane Netherlands  
  
# Adverse events CA (Source: event rates from NHR ablation analysis tool: Behandelgroep: Katheterablatie AF, Jaar: 2023)   
n\_CT\_CA <- 26 # cardiac tamponade  
n\_tot\_CT\_CA <- 6150  
p\_CT\_CA <- n\_CT\_CA/n\_tot\_CT\_CA  
n\_PP\_CA <- 25 # phrenicus paralysis  
n\_tot\_PP\_CA <- 6211  
p\_PP\_CA <- n\_PP\_CA/n\_tot\_PP\_CA   
n\_VC\_CA <- 40 # vascular complications  
n\_tot\_VC\_CA <- 6149  
p\_VC\_CA <- n\_VC\_CA/n\_tot\_VC\_CA   
  
#### COST INPUTS ####  
  
#### Health state costs and costs of interventions (Source: Vektis)  
df\_c\_states <- load(here::here("input", "c\_Vektis.RData"))   
  
#### Future medical costs  
# All costs using PAID tool, excluding costs for "Other heart diseases including pulmonary circulation" because we assume these are already included in the model in the cost inputs above.   
df\_FMC <- read.csv(here::here("input", "PAID\_AF\_FMC\_Unrelated\_Costs\_2025-02-26.csv"))   
colnames(df\_FMC) <- c("Age", "last\_y\_m", "last\_y\_f", "other\_y\_m", "other\_y\_f")  
df\_FMC[101:121,] <- df\_FMC[nrow(df\_FMC),] # repeat at age 99 for age 100-120  
df\_FMC$Age[101:121] <- seq(from = 100, to = 120, by = 1)  
df\_FMC <- df\_FMC %>%  
 pivot\_longer(  
 cols = c(last\_y\_m, last\_y\_f, other\_y\_m, other\_y\_f),  
 names\_to = c(".value", "Sex"),  
 names\_pattern = "(.\*)\_(.\*)$"  
 ) %>%  
 mutate(  
 Sex = ifelse(Sex == "f", 1, 0),  
 last\_y = as.numeric(last\_y),  
 other\_y = as.numeric(other\_y)  
 )  
  
#### Productivity costs inputs  
# Inputs are used in PrepareCosts to calculate productivity costs  
# Load the regression parameters and the variance covariance matrix of relationship between EQ-5D and productivity loss  
load(here::here("input", "betareg\_productivity\_AVATARAF\_NL.RData"))  
  
beta\_reg\_attendance\_intercept <- beta\_reg\_attendance$coefficients$mean[1]  
beta\_reg\_attendance\_SAF <- beta\_reg\_attendance$coefficients$mean[2]  
beta\_reg\_presenteeism\_intercept <- beta\_reg\_presenteeism$coefficients$mean[1]  
beta\_reg\_presenteeism\_SAF <- beta\_reg\_presenteeism$coefficients$mean[2]  
  
# Proportion of general population working, average hours per week and hourly wage  
# Calculations performed in PrepareCosts  
prop\_work\_M <- 0.681 # Source: CBS 2024 Netto arbeidsparticipatie tussen 45-75 jaar.  
prop\_work\_F <- 0.572 # Source: CBS 2024 Netto arbeidsparticipatie tussen 45-75 jaar.  
v\_hours\_per\_week\_M <- 35.9 # Source: "Werkzame beroepsbevolking; arbeidsduur" CBS 2024  
v\_hours\_per\_week\_F <- 27.9 # Source: "Werkzame beroepsbevolking; arbeidsduur" CBS 2024  
c\_hourly\_wage\_2022 <- 39.88 # Source: Dutch costing manual Euro 2022.   
  
#### Informal care costs   
# Informal care costs are based on a regression model of de Groot et al. (2023) that estimates informal care costs according to age and proximity to death  
# Regression model for probability of informal care (de Groot et al. 2023)  
params\_ic\_log\_intercept <- -1.451 # SE 0.220   
params\_ic\_log\_female <- 0.368 # SE 0.120   
params\_ic\_log\_age <- 0.054 # SE 0.008   
params\_ic\_log\_age2 <- 0.000 # SE 0.000   
params\_ic\_log\_T2D <- -0.061 # SE 0.015   
params\_ic\_hours\_intercept <- 0.497 # SE 0.198   
params\_ic\_hours\_female <- 0.112 # SE 0.106   
params\_ic\_hours\_age <- 0.019 # SE 0.005   
params\_ic\_hours\_T2D <- -0.034 # SE 0.011   
  
# Logistic regression for use of informal care  
# Coefficients on log-scale: intercept, gender (female = 1), age (age-centered at 70), age2, TTD (in years)  
params\_ic\_log <- c(params\_ic\_log\_intercept, params\_ic\_log\_female, params\_ic\_log\_age, params\_ic\_log\_age2, params\_ic\_log\_T2D)   
vcov\_ic\_log <- read.csv(here::here("input", "vcov\_ic\_log.csv"), sep = ",", header = FALSE)  
  
# Lineair regression for number of hours of informal care  
# Coefficients: intercept, gender (female = 1), age (age-centered at 70), TTD (in years)  
params\_ic\_hours <- c(params\_ic\_hours\_intercept, params\_ic\_hours\_female, params\_ic\_hours\_age, params\_ic\_hours\_T2D)  
vcov\_ic\_hours <- read.csv(here::here("input", "vcov\_ic\_hours.csv"), sep = ",", header = FALSE)  
  
# Inputs to calculate the average time to death and corresponding informal care costs of patients with the age of Van Den Dries et al. (i.e. 77 years)  
# Calculations performed in PrepareCosts  
LE\_M\_77 <- 10.3 # Life expectancy males of 77 years, Source: CBS  
LE\_F\_77 <- 11.9 # Life expectancy females of 77 years, Source: CBS  
c\_IC\_hr <- 18.8 # Hourly costs of informal care (Source: Kostenhandleiding, 2022 Euros)  
c\_IC\_77\_AF\_2yr <- 3296.45 #SD = 321.76, 24-month costs of AF patients in Van den Dries et al. (2023)   
c\_IC\_77\_AF\_2yr\_gamma <- gamma\_params(c\_IC\_77\_AF\_2yr, (321.76/sqrt(425)))  
  
#### UTILITIES INPUTS ####  
#### Disutilities AF and adverse events ####  
# Load utilities based on the EQ-5D-5L data in the AVATAR-AF trial using the Dutch tariff  
load(here::here("input", "glm\_AVATARAF\_NL.RData"))  
  
du\_SAF <- abs(unname((glm\_AVATARAF\_NL$coefficients[2]))) # Re-analysis of AVATAR-AF with EQ-5D-5L NL tariff based on Moss et al.   
  
# Assumption similar to Akerborg et al. (2012) and Reynolds et al. (2014)  
du\_CT\_mo <- 0.1 # Assumption disutility of 0.1 for 1 month, corrected to disutility for the cycle length in Effs  
du\_PP\_mo <- 0.1 # Assumption disutility of 0.1 for 1 month, corrected to disutility for the cycle length in Effs  
du\_VC\_mo <- 0.1 # Assumption disutility of 0.1 for 1 month, corrected to disutility for the cycle length in Effs  
  
#### General population utilities ####  
# Regression model general population by age and sex (based on Versteegh et al. 2016)  
load(here::here("input", "splines\_HRQoL.RData"))   
gen\_pop\_utility <- "fit"  
mod\_splines\_coef <- mod\_splines$coefficients  
  
#### STORE INPUT PARAMETERS ####  
# Create a vector of variable names  
v\_names\_params <- c("age", "sd\_age", "prop\_female", "v\_age", "v\_Sex", "prop\_parox",  
 "RR\_nai", "RR\_AAD\_exp\_parox", "RR\_AAD\_exp\_pers", "RR\_CA\_exp",   
 "prop\_AF\_CA\_nai", "prop\_AF\_CA\_AAD\_exp\_parox","prop\_AF\_CA\_AAD\_exp\_pers",   
 "prop\_AF\_CA\_CA\_exp\_1", "prop\_AF\_CA\_CA\_exp\_2",   
 "p\_SFAF\_CA\_SAF\_L1", "p\_SFAF\_CA\_SAF\_L2", "p\_SFAF\_CA\_SAF\_L3",   
 "HR\_EM", "p\_disc\_AAD\_nai", "p\_disc\_AAD\_exp",  
 "p\_CT\_CA", "p\_PP\_CA", "p\_VC\_CA",  
 "c\_AAD", "c\_CA\_DBC", "c\_d\_before\_CA", "c\_d\_after\_CA1", "c\_d\_after\_CA2", "c\_d\_SAF", "df\_FMC",   
 "prop\_work\_M", "prop\_work\_F", "v\_hours\_per\_week\_M", "v\_hours\_per\_week\_F",  
 "c\_hourly\_wage\_2022", "params\_ic\_log\_intercept", "params\_ic\_log\_female",   
 "params\_ic\_log\_age", "params\_ic\_log\_age2", 'params\_ic\_log\_T2D', "params\_ic\_hours\_intercept",  
 "params\_ic\_hours\_female", "params\_ic\_hours\_age", "params\_ic\_hours\_T2D",   
 "beta\_reg\_attendance\_intercept", "beta\_reg\_attendance\_SAF",   
 "beta\_reg\_presenteeism\_intercept", "beta\_reg\_presenteeism\_SAF",   
 "LE\_M\_77", "LE\_F\_77", "c\_IC\_hr", "c\_IC\_77\_AF\_2yr",  
 "du\_SAF", "du\_CT\_mo", "du\_PP\_mo", "du\_VC\_mo",   
 "gen\_pop\_utility", "mod\_splines\_coef"  
)  
  
# Store the parameters into a list  
l\_params <- mget(v\_names\_params)  
#View(l\_params)

# 03 Functions for microsimulation

## 03.1 df\_X: dataframe with individual level characteristics

Function to create the dataframe with patient characteristics

Create\_df\_X <- function(l\_params){  
 with((l\_params), {  
   
 set.seed(1)  
   
 # v\_age and v\_Sex are created in the previous section to enable variation in PSA  
 v\_Trt <- rep("AAD", n\_i) # Placeholder that will be overwritten in first section of MicroSim  
 v\_Line <- rep(1, n\_i)  
 v\_disc\_AAD <- rep(0, n\_i)  
 v\_CT\_CA <- rep(0, n\_i) # event counter for cardiac tamponade  
 v\_PP\_CA <- rep(0, n\_i) # event counter for phrenicus paralysis  
 v\_VC\_CA <- rep(0, n\_i) # event counter for vascular complications  
 v\_TH <- rep(0, n\_i) # Treatment history: 1 if current treatment is first month of CA (for side effects)  
 v\_AAD\_exp <- rep(0, n\_i) # Exposed to AAD  
 v\_CA\_exp\_1 <- rep(0, n\_i) # Exposed to CA  
 v\_CA\_exp\_2 <- rep(0, n\_i) # Exposed to at least 2 CAs  
 v\_death <- rep(0, n\_i) # Death  
   
   
 df\_X <- data.frame(ID = 1:n\_i, Age = v\_age, Age\_cl = v\_age, Age\_start = v\_age, Sex = v\_Sex,   
 curTrt = v\_Trt, Line = v\_Line,   
 disc\_AAD = v\_disc\_AAD,   
 CT\_CA = v\_CT\_CA, PP\_CA = v\_PP\_CA, VC\_CA = v\_VC\_CA,   
 TH = v\_TH, AAD\_exp = v\_AAD\_exp, CA\_exp\_1 = v\_CA\_exp\_1, CA\_exp\_2 = v\_CA\_exp\_2,   
 death = v\_death)  
   
 return(df\_X)  
 })  
}

## 03.2 Prepare probabilities

Function to prepare probabilities based on input parameters - Calculates weighted average of paroxysmal and persistent AF input parameters - Calculates probabilities for AAD based on CA and RR of AAD vs. CA - Converts probabilities to cycle length

PrepareProbs <- function(l\_params){  
   
 with((l\_params),{  
   
 # Calculate weighted average for prop\_AF\_CA\_AAD\_exp and RR\_AAD\_exp  
 prop\_AF\_CA\_AAD\_exp <- expit((logit(prop\_AF\_CA\_AAD\_exp\_parox)\*prop\_parox)+(logit(prop\_AF\_CA\_AAD\_exp\_pers)\*(1-prop\_parox)))   
   
 # Probability of recurrence of AF after AAD after 1 year   
 RR\_AAD\_exp <- exp((log(RR\_AAD\_exp\_parox)\*prop\_parox)+(log(RR\_AAD\_exp\_pers)\*(1-prop\_parox))) # weighted RR   
   
 # Based on proportion of recurrence of AF after CA multiplied with relative risk of AAD vs. CA  
 prop\_AF\_AAD\_nai <- prop\_AF\_CA\_nai\*RR\_nai   
 prop\_AF\_AAD\_AAD\_exp <- prop\_AF\_CA\_AAD\_exp\*RR\_AAD\_exp  
 prop\_AF\_AAD\_CA\_exp\_1 <- prop\_AF\_CA\_CA\_exp\_1\*RR\_CA\_exp  
 prop\_AF\_AAD\_CA\_exp\_2 <- prop\_AF\_CA\_CA\_exp\_2\*RR\_CA\_exp  
   
 # To prevent probabilities > 1 (not needed for deterministic analyses, only occurs in some iterations of the PSA)  
 prop\_AF\_CA\_nai <- ifelse(prop\_AF\_CA\_nai > 1, 1, prop\_AF\_CA\_nai)   
 prop\_AF\_CA\_AAD\_exp <- ifelse(prop\_AF\_CA\_AAD\_exp > 1, 1, prop\_AF\_CA\_AAD\_exp)   
 prop\_AF\_CA\_CA\_exp\_1 <- ifelse(prop\_AF\_CA\_CA\_exp\_1 > 1, 1, prop\_AF\_CA\_CA\_exp\_1)   
 prop\_AF\_CA\_CA\_exp\_2 <- ifelse(prop\_AF\_CA\_CA\_exp\_2 > 1, 1, prop\_AF\_CA\_CA\_exp\_2)   
 prop\_AF\_AAD\_nai <- ifelse(prop\_AF\_AAD\_nai > 1, 1, prop\_AF\_AAD\_nai )   
 prop\_AF\_AAD\_AAD\_exp <- ifelse(prop\_AF\_AAD\_AAD\_exp > 1, 1, prop\_AF\_AAD\_AAD\_exp )   
 prop\_AF\_AAD\_CA\_exp\_1 <- ifelse(prop\_AF\_AAD\_CA\_exp\_1 > 1, 1, prop\_AF\_AAD\_CA\_exp\_1)   
 prop\_AF\_AAD\_CA\_exp\_2 <- ifelse(prop\_AF\_AAD\_CA\_exp\_2 > 1, 1, prop\_AF\_AAD\_CA\_exp\_2)   
   
 # Convert annual probabilities to cycle length  
 p\_SFAF\_CA\_nai <- 1-convert\_probability\_to\_cl(prop\_AF\_CA\_nai, 1, cl)   
 p\_SFAF\_CA\_AAD\_exp <- 1-convert\_probability\_to\_cl(prop\_AF\_CA\_AAD\_exp, 1, cl)   
 p\_SFAF\_CA\_CA\_exp\_1 <- 1-convert\_probability\_to\_cl(prop\_AF\_CA\_CA\_exp\_1, 1, cl)   
 p\_SFAF\_CA\_CA\_exp\_2 <- 1-convert\_probability\_to\_cl(prop\_AF\_CA\_CA\_exp\_2, 1, cl)   
 p\_SFAF\_AAD\_nai <- 1-convert\_probability\_to\_cl(prop\_AF\_AAD\_nai, 1, cl)   
 p\_SFAF\_AAD\_AAD\_exp <- 1-convert\_probability\_to\_cl(prop\_AF\_AAD\_AAD\_exp, 1, cl)   
 p\_SFAF\_AAD\_CA\_exp\_1 <- 1-convert\_probability\_to\_cl(prop\_AF\_AAD\_CA\_exp\_1, 1, cl)   
 p\_SFAF\_AAD\_CA\_exp\_2 <- 1-convert\_probability\_to\_cl(prop\_AF\_AAD\_CA\_exp\_2, 1, cl)   
   
 l\_inputs\_probs\_names <- c("RR\_AAD\_exp", "p\_SFAF\_CA\_nai", "p\_SFAF\_CA\_AAD\_exp", "p\_SFAF\_CA\_CA\_exp\_1", "p\_SFAF\_CA\_CA\_exp\_2",  
 "p\_SFAF\_AAD\_nai", "p\_SFAF\_AAD\_AAD\_exp", "p\_SFAF\_AAD\_CA\_exp\_1", "p\_SFAF\_AAD\_CA\_exp\_2")  
 l\_inputs\_probs <- mget(l\_inputs\_probs\_names)  
   
 return(l\_inputs\_probs)  
 })  
}

## 03.3 Probability function

The function that updates the transition probabilities of every cycle is shown below.

Probs <- function(l\_params\_all, M\_t, df\_X, t) {   
 # Arguments:  
 # M\_t: health state occupied by individual i at cycle t (character variable)  
 # df\_X: data frame with individual characteristics data   
 # t: current cycle   
 # Returns:   
 # transition probabilities for that cycle  
   
 with((l\_params\_all),{  
   
 # Create matrix of state transition probabilities   
 m\_p\_t <- matrix(0, nrow = n\_states, ncol = n\_i)   
 rownames(m\_p\_t) <- v\_n # give the state names to the rows  
   
 # Look up probability of dying based on current age  
 df\_p\_D <- inner\_join(df\_X, df\_mort, by = c("Age", "Sex"))   
   
 # Multiply with HR for excess mortality and cycle length and convert back to probability  
 p\_D <- 1 - (1-df\_p\_D$p\_mort\_cl)^HR\_EM   
   
 # Success rates  
 p\_SAF\_SFAF <- NULL  
 p\_SAF\_SFAF[df\_X$curTrt == "AAD" & df\_X$AAD\_exp == 0 & df\_X$CA\_exp\_1 == 0] <- p\_SFAF\_AAD\_nai  
 p\_SAF\_SFAF[df\_X$curTrt == "AAD" & df\_X$AAD\_exp == 1 & df\_X$CA\_exp\_1 == 0] <- p\_SFAF\_AAD\_AAD\_exp  
 p\_SAF\_SFAF[df\_X$curTrt == "AAD" & df\_X$AAD\_exp == 0 & df\_X$CA\_exp\_1 == 1] <- p\_SFAF\_AAD\_CA\_exp\_1  
 p\_SAF\_SFAF[df\_X$curTrt == "AAD" & df\_X$AAD\_exp == 1 & df\_X$CA\_exp\_1 == 1] <- p\_SFAF\_AAD\_CA\_exp\_1  
 p\_SAF\_SFAF[df\_X$curTrt == "AAD" & df\_X$AAD\_exp == 0 & df\_X$CA\_exp\_2 == 1] <- p\_SFAF\_AAD\_CA\_exp\_2  
 p\_SAF\_SFAF[df\_X$curTrt == "AAD" & df\_X$AAD\_exp == 1 & df\_X$CA\_exp\_2 == 1] <- p\_SFAF\_AAD\_CA\_exp\_2  
   
 p\_SAF\_SFAF[df\_X$curTrt == "CA" & df\_X$AAD\_exp == 0 & df\_X$CA\_exp\_1 == 0] <- p\_SFAF\_CA\_nai  
 p\_SAF\_SFAF[df\_X$curTrt == "CA" & df\_X$AAD\_exp == 1 & df\_X$CA\_exp\_1 == 0] <- p\_SFAF\_CA\_AAD\_exp  
 p\_SAF\_SFAF[df\_X$curTrt == "CA" & df\_X$AAD\_exp == 0 & df\_X$CA\_exp\_1 == 1] <- p\_SFAF\_CA\_CA\_exp\_1  
 p\_SAF\_SFAF[df\_X$curTrt == "CA" & df\_X$AAD\_exp == 1 & df\_X$CA\_exp\_1 == 1] <- p\_SFAF\_CA\_CA\_exp\_1  
 p\_SAF\_SFAF[df\_X$curTrt == "CA" & df\_X$AAD\_exp == 0 & df\_X$CA\_exp\_2 == 1] <- p\_SFAF\_CA\_CA\_exp\_2  
 p\_SAF\_SFAF[df\_X$curTrt == "CA" & df\_X$AAD\_exp == 1 & df\_X$CA\_exp\_2 == 1] <- p\_SFAF\_CA\_CA\_exp\_2  
 p\_SAF\_SFAF[df\_X$curTrt == "no-treatment"] <- 0 # keep symptoms when no rhythm control treatment  
   
 # Recurrence rates at time t  
 p\_SFAF\_SAF <- NULL  
   
 p\_SFAF\_SAF[df\_X$curTrt == "AAD" & df\_X$AAD\_exp == 0 & df\_X$CA\_exp\_1 == 0] <- ifelse(p\_SFAF\_CA\_SAF\_L1[t]\*RR\_nai >1, 1, p\_SFAF\_CA\_SAF\_L1[t]\*RR\_nai )  
 p\_SFAF\_SAF[df\_X$curTrt == "AAD" & df\_X$AAD\_exp == 1 & df\_X$CA\_exp\_1 == 0] <- ifelse(p\_SFAF\_CA\_SAF\_L1[t]\*RR\_AAD\_exp >1, 1, p\_SFAF\_CA\_SAF\_L1[t]\*RR\_AAD\_exp)  
 p\_SFAF\_SAF[df\_X$curTrt == "AAD" & df\_X$AAD\_exp == 0 & df\_X$CA\_exp\_1 == 1] <- ifelse(p\_SFAF\_CA\_SAF\_L2[t]\*RR\_CA\_exp >1, 1, p\_SFAF\_CA\_SAF\_L2[t]\*RR\_CA\_exp )  
 p\_SFAF\_SAF[df\_X$curTrt == "AAD" & df\_X$AAD\_exp == 1 & df\_X$CA\_exp\_1 == 1] <- ifelse(p\_SFAF\_CA\_SAF\_L2[t]\*RR\_CA\_exp >1, 1, p\_SFAF\_CA\_SAF\_L2[t]\*RR\_CA\_exp )  
 p\_SFAF\_SAF[df\_X$curTrt == "AAD" & df\_X$AAD\_exp == 0 & df\_X$CA\_exp\_2 == 1] <- ifelse(p\_SFAF\_CA\_SAF\_L3[t]\*RR\_CA\_exp >1, 1, p\_SFAF\_CA\_SAF\_L3[t]\*RR\_CA\_exp )  
 p\_SFAF\_SAF[df\_X$curTrt == "AAD" & df\_X$AAD\_exp == 1 & df\_X$CA\_exp\_2 == 1] <- ifelse(p\_SFAF\_CA\_SAF\_L3[t]\*RR\_CA\_exp >1, 1, p\_SFAF\_CA\_SAF\_L3[t]\*RR\_CA\_exp )  
   
 p\_SFAF\_SAF[df\_X$curTrt == "CA" & df\_X$AAD\_exp == 0 & df\_X$CA\_exp\_1 == 0] <- p\_SFAF\_CA\_SAF\_L1[t]  
 p\_SFAF\_SAF[df\_X$curTrt == "CA" & df\_X$AAD\_exp == 1 & df\_X$CA\_exp\_1 == 0] <- p\_SFAF\_CA\_SAF\_L1[t]  
 p\_SFAF\_SAF[df\_X$curTrt == "CA" & df\_X$AAD\_exp == 0 & df\_X$CA\_exp\_1 == 1] <- p\_SFAF\_CA\_SAF\_L2[t]  
 p\_SFAF\_SAF[df\_X$curTrt == "CA" & df\_X$AAD\_exp == 1 & df\_X$CA\_exp\_1 == 1] <- p\_SFAF\_CA\_SAF\_L2[t]  
 p\_SFAF\_SAF[df\_X$curTrt == "CA" & df\_X$AAD\_exp == 0 & df\_X$CA\_exp\_2 == 1] <- p\_SFAF\_CA\_SAF\_L3[t]  
 p\_SFAF\_SAF[df\_X$curTrt == "CA" & df\_X$AAD\_exp == 1 & df\_X$CA\_exp\_2 == 1] <- p\_SFAF\_CA\_SAF\_L3[t]  
  
 p\_SFAF\_SAF[df\_X$curTrt == "no-treatment"] <- 1-p\_D[df\_X$curTrt == "no-treatment"] # get symptoms when no rhythm control treatment  
  
 # Fill the transition probability matrix with the appropriate probabilities  
 m\_p\_t[, M\_t == "SFAF"] <- rbind((1-p\_SFAF\_SAF[M\_t == "SFAF"]-p\_D[M\_t == "SFAF"]),  
 p\_SFAF\_SAF[M\_t == "SFAF"],   
 p\_D[M\_t == "SFAF"])  
 m\_p\_t[, M\_t == "SAF"] <- rbind(p\_SAF\_SFAF[M\_t == "SAF"],   
 1-p\_SAF\_SFAF[M\_t == "SAF"]-p\_D[M\_t == "SAF"],   
 p\_D[M\_t == "SAF"])  
 m\_p\_t[, M\_t == "D"] <- rbind(0, 0, 1)  
   
 if(any(is.na(m\_p\_t))) {  
 warning("NA values found in transition probability matrix")  
 print(which(is.na(m\_p\_t), arr.ind = TRUE))  
 }  
   
 return(t(m\_p\_t))  
 }) # End of with l\_params  
}

## 03.4 Prepare costs

PrepareCosts <- function(l\_params){  
   
 with((l\_params),{  
   
 #### Health state costs #### (Source: Vektis)  
 c\_before\_CA <- adjust\_inflation(c\_d\_before\_CA, "2021")  
 c\_after\_CA <- adjust\_inflation(c\_d\_after\_CA1, "2021") + adjust\_inflation(c\_d\_after\_CA2, "2021")  
 c\_d\_SAF <- adjust\_inflation(c\_d\_SAF, "2021")  
   
 #### Intervention costs ####  
 # Based on average AAD costs in patients with symptoms of AF (12-6 months before CA)  
 c\_AAD <- adjust\_inflation(c\_AAD, "2021")  
   
 # Intervention costs + additional health care consumption 6 months before and 12 months after CA  
 c\_CA <- adjust\_inflation(c\_CA\_DBC, "2021")+c\_before\_CA+c\_after\_CA   
 c\_CA <- unname(c\_CA)  
   
 #### Future medical costs ####  
 df\_FMC$other\_y <- df\_FMC$other\_y\*cl # Adapt to the cycle length  
 df\_FMC$last\_y <- adjust\_inflation(df\_FMC$last\_y, "2017")  
 df\_FMC$other\_y <- adjust\_inflation(df\_FMC$other\_y, "2017")  
 df\_FMC <- as.data.frame(df\_FMC)  
   
 #### Productivity costs ####  
 # Productivity costs consist of three components in this model, recovery time after CA (1 week), work attendance and presenteeism  
   
 # Work attendance and presenteeism are estimated with regression models from the EQ-5D-5L to 3L converted health states, age (work attendance) and sex (presenteeism) (Source: Krol et al. 2014). We used the AVATAR-AF data to estimate work attendance and presenteeism for each AF patient in the trial. Then we used a beta-regression to estimate how symptom status influenced work attendance and presenteeism.  
   
 # Probability of work attendance based on symptoms. Source: AVATAR-AF and Krol et al. (2014)  
 p\_SFAF\_workattendance <- exp(beta\_reg\_attendance\_intercept)/(1+exp(beta\_reg\_attendance\_intercept))  
 p\_SAF\_workattendance <- exp((beta\_reg\_attendance\_intercept+beta\_reg\_attendance\_SAF))/  
 (1+exp(beta\_reg\_attendance\_intercept+beta\_reg\_attendance\_SAF))  
   
 # Proportion at work based on general population corrected for age and sex distribution in AF population  
 prop\_work <- (prop\_work\_M\*(1-prop\_female)) + prop\_work\_F\*prop\_female  
   
 # Correct proportion at work for work attendance of AF patients with/without symptoms  
 prop\_work\_SAF <- prop\_work\*p\_SAF\_workattendance # Proportion of patients at work with AF symptoms  
 prop\_work\_SFAF <- prop\_work\*p\_SFAF\_workattendance # Proportion of patients at work without AF symptoms  
   
 # Probability of presenteeism (i.e. productivity at work)  
 p\_SFAF\_presenteeism <- exp(beta\_reg\_presenteeism\_intercept)/(1+exp(beta\_reg\_presenteeism\_intercept))  
 p\_SAF\_presenteeism <- exp((beta\_reg\_presenteeism\_intercept+beta\_reg\_presenteeism\_SAF))/  
 (1+exp(beta\_reg\_presenteeism\_intercept+beta\_reg\_presenteeism\_SAF))  
   
 # Productivity costs based on average work duration and wage  
 v\_hours\_per\_week <- v\_hours\_per\_week\_F\*prop\_female+(v\_hours\_per\_week\_M\*(1-prop\_female))   
 v\_hours\_per\_cycle <- cl\*(365.25/7)\*v\_hours\_per\_week # Work hours in a cycle  
 c\_hourly\_wage <- adjust\_inflation(c\_hourly\_wage\_2022, "2022")   
 c\_prod\_cycle <- v\_hours\_per\_cycle\*c\_hourly\_wage  
   
 # Total productivity: work attendance corrected for presenteeism multiplied with costs  
 c\_prod\_SAF\_tot <- prop\_work\_SAF\*p\_SAF\_presenteeism\*c\_prod\_cycle # Total productivity costs patients with AF symptoms  
 c\_prod\_SFAF\_tot <- prop\_work\_SFAF\*c\_prod\_cycle\*p\_SFAF\_presenteeism # Total productivity costs patients without AF symptoms  
 c\_prod\_SAF <- unname(c\_prod\_SFAF\_tot-c\_prod\_SAF\_tot) # Total productivity costs attributable to having AF symptoms  
   
 # Productivity costs of recovery after CA: 1 week absent from work  
 c\_prod\_CA\_recovery <- v\_hours\_per\_week\*c\_hourly\_wage\*prop\_work\_SAF  
   
 #### Informal care costs ####  
 # Informal care costs are based on a regression model of de Groot et al. (2023) that estimates informal care costs according to age and proximity to death  
 # Weighted average life expectancy AF gender distribution, life expectancy is equal to time to death  
 T2D <- (LE\_M\_77\*(1-prop\_female))+(LE\_F\_77\*prop\_female)   
   
 # Calculate informal care costs for a 77-year old from the general population   
 # Age is centered at the mean in the analysis and 70 is the mean age in de Groot et al. (2023)   
 p\_IC\_log <- params\_ic\_log\_intercept + params\_ic\_log\_female\*prop\_female + params\_ic\_log\_age\*(77-70) +   
 params\_ic\_log\_age2\*((77-70)\*(77-70)) + params\_ic\_log\_T2D\*T2D # Propotion use of informal care  
 p\_IC <- exp(p\_IC\_log)/(1+exp(p\_IC\_log)) # Convert to probability  
 v\_hr\_IC\_log <- params\_ic\_hours\_intercept + params\_ic\_hours\_female\*prop\_female + params\_ic\_hours\_age\*(77-70) + params\_ic\_hours\_T2D\*T2D   
 v\_hr\_day\_IC <- exp(v\_hr\_IC\_log) # Convert from log scale to hours per day  
 c\_IC\_77\_cl <- p\_IC\*(v\_hr\_day\_IC\*365.25\*cl)\*c\_IC\_hr # Total costs of informal care in 77-year old in a cycle  
 c\_IC\_77\_AF\_cl <- (c\_IC\_77\_AF\_2yr/2)\*cl # correct for cycle length  
 c\_IC\_SAF <- c\_IC\_77\_AF\_cl-c\_IC\_77\_cl # Informal care costs attributable to symptomatic AF  
   
 l\_inputs\_costs\_names <- c("c\_d\_SAF", "c\_AAD", "c\_CA", "df\_FMC", "c\_prod\_SAF", "c\_prod\_CA\_recovery", "c\_IC\_SAF")  
   
 l\_inputs\_costs <- mget(l\_inputs\_costs\_names)  
   
 return(l\_inputs\_costs)  
 })  
}

## 03.5 Cost function

The Costs function estimates the costs at every cycle.

Costs <- function (l\_params\_all, M\_t, df\_X) {  
 # M\_t: health state occupied by individual i at cycle t (character variable)  
   
 with((l\_params\_all),{  
   
 # Objects for future medical costs  
 c\_FMC <- inner\_join(df\_X, df\_FMC, by = c("Age", "Sex")) # Look up future medical costs based on current age and sex  
   
 # Assign costs based on health states (informal care costs will be added at the end of the MicroSim)  
 c\_t <- NULL   
  
 c\_t[M\_t == "SFAF" & df\_X$curTrt == "AAD"] <- c\_AAD + c\_FMC$other\_y[M\_t == "SFAF" & df\_X$curTrt == "AAD"]   
   
 c\_t[M\_t == "SFAF" & df\_X$curTrt == "CA"] <- c\_CA \* df\_X$TH[M\_t == "SFAF" & df\_X$curTrt == "CA"] +   
 c\_FMC$other\_y[M\_t == "SFAF" & df\_X$curTrt == "CA"] +  
 (c\_prod\_CA\_recovery \* df\_X$TH[M\_t == "SFAF" & df\_X$curTrt == "CA"]) \*   
 (df\_X$Age[M\_t == "SFAF" & df\_X$curTrt == "CA"]<68)  
   
 c\_t[M\_t == "SFAF" & df\_X$curTrt == "no-treatment"] <- c\_FMC$other\_y[M\_t == "SFAF" & df\_X$curTrt == "no-treatment"]   
   
 c\_t[M\_t == "SAF" & df\_X$curTrt == "AAD"] <- c\_AAD + c\_d\_SAF + c\_FMC$other\_y[M\_t == "SAF" & df\_X$curTrt == "AAD"] +  
 c\_prod\_SAF\*(df\_X$Age[M\_t == "SAF" & df\_X$curTrt == "AAD"]<68)   
   
 c\_t[M\_t == "SAF" & df\_X$curTrt == "CA"] <- c\_CA + c\_d\_SAF + c\_FMC$other\_y[M\_t == "SAF" & df\_X$curTrt == "CA"] +  
 c\_prod\_SAF\*(df\_X$Age[M\_t == "SAF" & df\_X$curTrt == "CA"]<68) +   
 (c\_prod\_CA\_recovery \* df\_X$TH[M\_t == "SAF" & df\_X$curTrt == "CA"]) \*   
 (df\_X$Age[M\_t == "SAF" & df\_X$curTrt == "CA"]<68)   
   
 c\_t[M\_t == "SAF" & df\_X$curTrt == "no-treatment"] <- c\_d\_SAF + c\_FMC$other\_y[M\_t == "SAF" & df\_X$curTrt == "no-treatment"] +  
 c\_prod\_SAF\*(df\_X$Age[M\_t == "SAF" & df\_X$curTrt == "no-treatment"]<68)   
   
 c\_t[M\_t == "D"] <- c\_FMC$last\_y[M\_t == "D"]\*df\_X$death[M\_t == "D"]  
   
 return(c\_t) # return the costs  
   
 }) # end of with l\_params\_all  
}

## 03.6 Health outcome function

The Effs function to update the utilities at every cycle.

Effs <- function (l\_params\_all, M\_t, df\_X, t, cl = cl) {  
 # M\_t: health state occupied by individual i at cycle t (character variable)  
 # df\_X: data frame with individual characteristics data   
   
 with((l\_params\_all),{  
   
 # Determine distutilites of adverse events after CA  
 du\_CT <- du\_CT\_mo\*cl # Assumption disutility of 0.1 for 1 month, corrected to disutility for the cycle length  
 du\_PP <- du\_PP\_mo\*cl # Assumption disutility of 0.1 for 1 month, corrected to disutility for the cycle length  
 du\_VC <- du\_VC\_mo\*cl # Assumption disutility of 0.1 for 1 month, corrected to disutility for the cycle length  
   
 # Create vector with total disutility per patient based on df\_X  
 du\_AE\_CA <- NULL  
 du\_AE\_CA <- (df\_X$CT\_CA\*du\_CT)+(df\_X$PP\_CA\*du\_PP)+(df\_X$VC\_CA\*du\_VC)   
   
 # Determine baseline utility based on general population estimates  
 mod\_splines$coefficients <- mod\_splines\_coef # included for use in PSA  
   
 if(gen\_pop\_utility == "fit"){ # For use in base case  
 u\_t <- unname(predict(mod\_splines, newdata = df\_X))   
 }  
   
 if(gen\_pop\_utility == "lwr"){ # For use in OWSA  
 u\_t <- unname(predict(mod\_splines, newdata = df\_X, interval = "confidence")[, "lwr"])   
 }  
   
 if(gen\_pop\_utility == "upr"){ # For use in OWSA  
 u\_t <- unname(predict(mod\_splines, newdata = df\_X, interval = "confidence")[, "upr"])   
 }  
   
 # Assign utilities to health states  
 u\_t[M\_t == "SFAF"] <- u\_t[M\_t == "SFAF"] - du\_AE\_CA[M\_t == "SFAF"]   
 u\_t[M\_t == "SAF"] <- u\_t[M\_t == "SAF"] - du\_AE\_CA[M\_t == "SAF"] - du\_SAF   
 u\_t[M\_t == "D"] <- 0  
   
 QALYs <- u\_t \* cl # calculate the QALYs during cycle t  
   
 return(QALYs) # return the QALYs  
   
 }) # end of with l\_params\_all  
}

## 03.7 Microsimulation

MicroSim <- function(l\_params, n\_i, df\_X, TRT1 = TRT1, TRT2 = TRT2, TRT3 = TRT3,   
 TRT4 = TRT4, TRT5 = TRT5, TRT6 = TRT6, seed = 1) {  
   
 # Arguments:   
 # n\_i: number of individuals  
 # df\_X data frame with individual characteristics data   
 # TRT1-3: The treatments in the sequence  
 # seed: default is 1  
   
 #### Set up starting values ####  
 df\_X <- Create\_df\_X(l\_params) # Create dataframe with patient characteristics  
 df\_X$curTrt <- TRT1 # Assign the first treatment to all patients  
 v\_M\_init <- rep("SAF", n\_i) # All patients start with AF symptoms   
 df\_X$TH <- ifelse(df\_X$curTrt == "CA", 1, 0) # If a patient starts on CA, treatment history (TH) is set to 1   
   
 # Calculate input parameters and combine them in l\_params\_all  
 l\_inputs\_Probs <- PrepareProbs(l\_params)  
 l\_inputs\_Costs <- PrepareCosts(l\_params)  
 l\_params\_all <- c(l\_params, l\_inputs\_Probs, l\_inputs\_Costs)  
 l\_params\_all <- l\_params\_all[!duplicated(names(l\_params\_all), fromLast = T)]  
  
 with((l\_params\_all), {  
 set.seed(seed) # set the seed  
 n\_states <- length(v\_n) # the number of health states  
   
 #### Create matrices ####  
 # create matrices with number of rows equal to the n\_i, the number of columns equal to n\_t   
 # (the initial state and all the n\_t cycles)  
 # m\_M is used to store the health state information over time for every individual  
 # m\_C is used to store the costs information over time for every individual  
 # m\_E is used to store the effects information over time for every individual  
 # m\_L is used to store the line information over time for every individual  
 # m\_TH is used to store the treatment history of CA over time for every individual  
 # m\_AE is used to store the adverse events after CA over time for every individual  
 # m\_D is used to store the death status over time for every individual; used to calculate future medical costs  
 # m\_C\_IC\_cl is used to store the informal care costs over time for every individual  
 # m\_curTrt is used to store the current treatment over time for every individual  
 # m\_Age is used to store the current age over time for every individual  
   
 m\_M <- m\_C <- m\_E <- m\_L <- m\_TH <- m\_AE <- m\_D <- m\_C\_IC\_cl <- m\_curTrt <- m\_Age <- matrix(nrow = n\_i, ncol = n\_t + 1,   
 dimnames = list(paste("ind" , 1:n\_i, sep = " "),   
 paste("cycle", 0:n\_t, sep = " ")))   
   
 m\_M [, 1] <- v\_M\_init # initial health state at cycle 0 for individual i  
 m\_TH[, 1] <- df\_X$TH # initial treatment history at cycle 0 for individual i  
 m\_L [, 1] <- rep(1, n\_i) # initial line at cycle 0 for individual i  
 m\_D [, 1] <- df\_X$death # initial death status at cycle 0 for individual i  
 m\_C\_IC\_cl[, 1] <- rep(0, n\_i) # initial costs of informal care  
 m\_curTrt[, 1] <- df\_X$curTrt # initial treatment  
 m\_Age[, 1] <- df\_X$Age # initial age  
  
 #### Cycle 0 ####  
 # Costs and QALYs in cycle 1  
 m\_C[, 1] <- Costs(l\_params\_all, m\_M[, 1], df\_X)   
 m\_E[, 1] <- Effs (l\_params\_all, m\_M[, 1], df\_X, t = 1, cl = cl)   
   
 #### Start loop cycle 1 to n\_t ####  
 # Open a loop for time running cycles 1 to n\_t   
 for (t in 1:n\_t) {  
   
 # To remove variability due to random draw procedure (seed) but keep variation between cycles (+ t)  
 set.seed(seed + t)  
   
 #### Switch health states ####  
 # Calculate the transition probabilities for the cycle based on health state t  
 m\_P <- Probs(l\_params\_all, m\_M[, t], df\_X, t)   
   
 # Sample the current health state based on the transition probabilities and store that state in matrix m\_M   
 m\_M[, t + 1] <- samplev(m\_P, 1)   
   
 #### Adverse events ####  
 # Discontinue AAD due to adverse events  
 df\_X$disc\_AAD[df\_X$curTrt == "AAD" & df\_X$AAD\_exp == 0] <- rbinom(sum(df\_X$curTrt == "AAD" & df\_X$AAD\_exp == 0), 1, p\_disc\_AAD\_nai)  
 df\_X$disc\_AAD[df\_X$curTrt == "AAD" & df\_X$AAD\_exp == 1] <- rbinom(sum(df\_X$curTrt == "AAD" & df\_X$AAD\_exp == 1), 1, p\_disc\_AAD\_exp)  
   
 # Adverse events during the first cycle after CA   
 df\_X$PP\_CA[df\_X$curTrt == "CA" & df\_X$TH == 1] <- rbinom(sum(df\_X$curTrt == "CA" & df\_X$TH == 1), 1, p\_PP\_CA)  
 df\_X$CT\_CA[df\_X$curTrt == "CA" & df\_X$TH == 1] <- rbinom(sum(df\_X$curTrt == "CA" & df\_X$TH == 1), 1, p\_CT\_CA)  
 df\_X$VC\_CA[df\_X$curTrt == "CA" & df\_X$TH == 1] <- rbinom(sum(df\_X$curTrt == "CA" & df\_X$TH == 1), 1, p\_VC\_CA)  
   
 m\_AE[, t+ 1] <- df\_X$PP\_CA + df\_X$CT\_CA +df\_X$VC\_CA   
   
 #### Switching lines ####  
 # Switch lines when symptoms of AF or discontinuation of AAD  
 # If maximum number of lines has been reached, switch to no rhythm control treatment  
 df\_X$Line[m\_M[ , t + 1] == "SFAF" & df\_X$disc\_AAD == 0] <- df\_X$Line[m\_M[ , t + 1] == "SFAF" & df\_X$disc\_AAD == 0]   
 df\_X$Line[m\_M[ , t + 1] == "SFAF" & df\_X$disc\_AAD == 1 & df\_X$Line >= 6] <- 555 # no rhythm control treatment   
 df\_X$Line[m\_M[ , t + 1] == "SFAF" & df\_X$disc\_AAD == 1 & df\_X$Line < 6] <- df\_X$Line[m\_M[ , t + 1] == "SFAF"& df\_X$disc\_AAD == 1 & df\_X$Line < 6] + 1   
 df\_X$Line[m\_M[ , t + 1] == "SAF" & df\_X$Line >= 6] <- 555 # no rhythm control treatment  
 df\_X$Line[m\_M[ , t + 1] == "SAF" & df\_X$Line < 6] <- df\_X$Line[m\_M[ , t + 1] == "SAF" & df\_X$Line < 6] + 1  
 df\_X$Line[m\_M[ , t + 1] == "D"] <- 999 # dead  
   
 # Update matrix with current line number  
 m\_L[ , t + 1] <- df\_X$Line  
   
 # Update current treatment for the next cycle  
 df\_X$curTrt[df\_X$Line == 1] <- TRT1  
 df\_X$curTrt[df\_X$Line == 2] <- TRT2  
 df\_X$curTrt[df\_X$Line == 3] <- TRT3  
 df\_X$curTrt[df\_X$Line == 4] <- TRT4  
 df\_X$curTrt[df\_X$Line == 5] <- TRT5  
 df\_X$curTrt[df\_X$Line == 6] <- TRT6  
 df\_X$curTrt[df\_X$Line == 555] <- "no-treatment"  
 df\_X$curTrt[df\_X$Line == 999] <- "D"  
   
 m\_curTrt[, t + 1] <- df\_X$curTrt  
   
 # Update the evaluation of first month of ablation  
 df\_X$TH <- ifelse(df\_X$curTrt == "CA" & m\_L[,t] != m\_L[,t+1], 1, 0)  
 m\_TH[, t + 1] <- df\_X$TH  
   
 # Update variables for AAD or CA exposure  
 # Exposed to AAD  
 df\_X$AAD\_exp[TRT1 == "AAD" & df\_X$Line > 1] <- 1  
 df\_X$AAD\_exp[TRT2 == "AAD" & df\_X$Line > 2] <- 1  
 df\_X$AAD\_exp[TRT3 == "AAD" & df\_X$Line > 3] <- 1  
 df\_X$AAD\_exp[TRT4 == "AAD" & df\_X$Line > 4] <- 1  
 df\_X$AAD\_exp[TRT5 == "AAD" & df\_X$Line > 5] <- 1  
  
 # Exposed to CA at least once  
 df\_X$CA\_exp\_1[TRT1 == "CA" & df\_X$Line > 1] <- 1  
 df\_X$CA\_exp\_1[TRT2 == "CA" & df\_X$Line > 2] <- 1  
 df\_X$CA\_exp\_1[TRT3 == "CA" & df\_X$Line > 3] <- 1  
 df\_X$CA\_exp\_1[TRT4 == "CA" & df\_X$Line > 4] <- 1  
 df\_X$CA\_exp\_1[TRT5 == "CA" & df\_X$Line > 5] <- 1  
  
 # Exposed to CA at least twice  
 df\_X$CA\_exp\_2[TRT1 == "CA" & TRT2 == "CA" & df\_X$Line > 2] <- 1  
 df\_X$CA\_exp\_2[TRT1 == "CA" & TRT3 == "CA" & df\_X$Line > 3] <- 1  
 df\_X$CA\_exp\_2[TRT1 == "CA" & TRT4 == "CA" & df\_X$Line > 4] <- 1  
 df\_X$CA\_exp\_2[TRT1 == "CA" & TRT5 == "CA" & df\_X$Line > 5] <- 1  
 df\_X$CA\_exp\_2[TRT2 == "CA" & TRT3 == "CA" & df\_X$Line > 3] <- 1  
 df\_X$CA\_exp\_2[TRT2 == "CA" & TRT4 == "CA" & df\_X$Line > 4] <- 1  
 df\_X$CA\_exp\_2[TRT2 == "CA" & TRT5 == "CA" & df\_X$Line > 5] <- 1  
 df\_X$CA\_exp\_2[TRT3 == "CA" & TRT4 == "CA" & df\_X$Line > 4] <- 1  
 df\_X$CA\_exp\_2[TRT3 == "CA" & TRT5 == "CA" & df\_X$Line > 5] <- 1  
 df\_X$CA\_exp\_2[TRT4 == "CA" & TRT5 == "CA" & df\_X$Line > 5] <- 1  
  
 # Set adverse events status back to zero for everyone   
 df\_X$disc\_AAD <- df\_X$CT\_CA <- df\_X$PP\_CA <- df\_X$VC\_CA <- 0   
   
 # Update the age of individuals that are alive  
 df\_X$Age\_cl[m\_M[, t + 1] != "D"] <- df\_X$Age\_cl[m\_M[, t + 1] != "D"] + cl  
 df\_X$Age[m\_M[, t + 1] != "D"] <- round\_age(df\_X$Age\_cl[m\_M[, t + 1] != "D"]) # rounded for use with background mortality and FMC  
   
 m\_Age[, t + 1] <- df\_X$Age  
   
 # Update death status in m\_D and in df\_X to capture the costs of last year of life in the Costs function  
 m\_D[, t + 1] <- ifelse(m\_M[, t + 1] == "D" , 1, 0)  
 df\_X$death <- ifelse(m\_D[, t] == 0 & m\_D[, t + 1] == 1, 1, 0) # i.e. only 1 if not dead in previous cycle  
   
 #### Calculate costs and QALYs ####  
 # Calculate costs per individual during cycle t + 1  
 m\_C[, t + 1] <- Costs(l\_params\_all, m\_M[, t + 1], df\_X)   
   
 # Calculate QALYs per individual during cycle t + 1  
 m\_E[, t + 1] <- Effs(l\_params\_all, m\_M[, t + 1], df\_X, t, cl = cl)   
   
 # Display simulation progress  
 if(t/(n\_t/10) == round(t/(n\_t/10), 0)) { # display progress every 10%  
 cat('\r', paste(t/n\_t \* 100, "% done", sep = " "))  
 }  
   
 } # close the loop for the time points   
   
 #### Calculate informal care costs based on T2D ####  
 # Calculate the time to death per individual  
 T2D\_data\_temp <- ifelse(m\_M == "D", 0, 1) # Replace all 'A' for 1 and all 'D' for 0  
 T2D\_data <- unname(rowSums(T2D\_data\_temp)\*cl) # Calculate the sums of every row i.e. the time to death per individual, expressed in years  
 m\_C\_IC\_a <- matrix(nrow = n\_i, ncol = (n\_t + 1)\*cl)  
 m\_C\_IC\_a[, 1] <- rep(0, n\_i) # Matrix to capture informal care costs with time to death  
   
 # Start loop over patients  
 for(i in 1:n\_i){   
 T2D <- T2D\_data[i] # determine time to death of individual i  
 # Start loop over time to death for individual i  
 for(t in 1:T2D){ # loop over start simulation until death of individual i  
   
 # Calculate informal care costs for individual i at time t  
 p\_care\_use <- l\_params$params\_ic\_log\_intercept + l\_params$params\_ic\_log\_female\*df\_X$Sex[i] +  
 l\_params$params\_ic\_log\_age\*((df\_X$Age\_start[i]+t)-70) +   
 l\_params$params\_ic\_log\_age2\*(((df\_X$Age\_start[i]+t)-70)\*((df\_X$Age\_start[i]+t)-70)) + l\_params$params\_ic\_log\_T2D\*(T2D-t)  
 v\_hour\_care <- l\_params$params\_ic\_hours\_intercept + l\_params$params\_ic\_hours\_female\*df\_X$Sex[i] +  
 l\_params$params\_ic\_hours\_age\*((df\_X$Age\_start[i]+t)-70) + l\_params$params\_ic\_hours\_T2D\*(T2D-t) #estimates hours per day  
 m\_C\_IC\_a[i,t] <- (exp(p\_care\_use)/(1+exp(p\_care\_use)))\*exp(v\_hour\_care)\*c\_IC\_hr\*365.25 # matrix with annual costs  
 }  
 }  
   
 # Adjust the caregiver costs annual matrix to cycle length  
 for (h in 1:((n\_t + 1)\*cl)) {  
 # Each original column value is split evenly between two new columns  
 m\_C\_IC\_cl[, 2\*h-1] <- m\_C\_IC\_a[, h]\*cl  
 m\_C\_IC\_cl[, 2\*h] <- m\_C\_IC\_a[, h]\*cl  
 }  
   
 # Adjust the caregiver matrix for those with and without symptoms  
 m\_C\_IC\_cl[m\_M == "SAF"] <- m\_C\_IC\_cl[m\_M == "SAF"] + c\_IC\_SAF # Add the AF specific caregiver burden costs  
   
 #### Calculate and discount lifetime costs and effects ####  
 m\_C\_IC\_cl <- ifelse(is.na(m\_C\_IC\_cl), 0, m\_C\_IC\_cl) # replace NAs with zero in the cycles where patients are dead  
 m\_C <- m\_C + m\_C\_IC\_cl # combine regular costs with informal care costs  
 m\_LY <- ifelse(m\_M=="D", 0, cl) # undiscounted life years (i.e. not corrected for quality of life)  
   
 tc <- m\_C %\*% (v\_dwc \* v\_wcc) # total discounted cost per individual  
 te <- m\_E %\*% (v\_dwe \* v\_wcc) # total discounted QALYs per individual   
 tLY <- m\_LY %\*% (v\_dwe \* v\_wcc) # total discounted LYs per individual   
 tLY\_undisc <- m\_LY %\*% (v\_wcc) # total undiscounted LYs per individual   
   
 tc\_hat <- mean(tc) # average discounted cost   
 te\_hat <- mean(te) # average discounted QALYs  
 tLY\_hat <- mean(tLY) # average disounted LYs  
 tLY\_undisc\_hat <- mean(tLY\_undisc) # average undisounted LYs  
   
   
 #### Store the results from the simulation in a list ####  
 if(PSA == F){  
   
 # Determine time on treatment in each line  
 m\_ToT <- m\_L   
 m\_ToT[m\_M == "D"] <- NA # if you are dead, remove treatment line  
 m\_ToT <- m\_ToT[,-1] # remove cycle 0  
   
 # Determine proportion on treatment line  
 df\_PoT <- data.frame(p\_L1 = rep(0, n\_i),  
 p\_L2 = rep(0, n\_i),  
 p\_L3 = rep(0, n\_i),  
 p\_L4 = rep(0, n\_i),  
 p\_L5 = rep(0, n\_i),  
 p\_L6 = rep(0, n\_i),   
 p\_noTrt = rep(0, n\_i))  
   
 df\_PoT[ , 1] <- as.numeric(ifelse(rowSums(m\_ToT=="1", na.rm = T)==0, NA, rowSums(m\_ToT=="1", na.rm = T)))   
 df\_PoT[ , 2] <- as.numeric(ifelse(rowSums(m\_ToT=="2", na.rm = T)==0, NA, rowSums(m\_ToT=="2", na.rm = T)))  
 df\_PoT[ , 3] <- as.numeric(ifelse(rowSums(m\_ToT=="3", na.rm = T)==0, NA, rowSums(m\_ToT=="3", na.rm = T)))  
 df\_PoT[ , 4] <- as.numeric(ifelse(rowSums(m\_ToT=="4", na.rm = T)==0, NA, rowSums(m\_ToT=="4", na.rm = T)))  
 df\_PoT[ , 5] <- as.numeric(ifelse(rowSums(m\_ToT=="5", na.rm = T)==0, NA, rowSums(m\_ToT=="5", na.rm = T)))  
 df\_PoT[ , 6] <- as.numeric(ifelse(rowSums(m\_ToT=="6", na.rm = T)==0, NA, rowSums(m\_ToT=="6", na.rm = T)))  
 df\_PoT[ , 7] <- as.numeric(ifelse(rowSums(m\_ToT=="555", na.rm = T)==0, NA, rowSums(m\_ToT=="555", na.rm = T)))  
   
 t\_L1 <- mean(df\_PoT[, 1], na.rm = T)\*cl   
 t\_L2 <- mean(df\_PoT[, 2], na.rm = T)\*cl   
 t\_L3 <- mean(df\_PoT[, 3], na.rm = T)\*cl   
 t\_L4 <- mean(df\_PoT[, 4], na.rm = T)\*cl   
 t\_L5 <- mean(df\_PoT[, 5], na.rm = T)\*cl   
 t\_L6 <- mean(df\_PoT[, 6], na.rm = T)\*cl   
 t\_noTrt <- mean(df\_PoT[, 7], na.rm = T)\*cl   
   
 # Proportion receiving treatment  
 p\_L1 <- 100 #everybody starts on line 1  
 p\_L2 <- (sum(!is.na(df\_PoT[, 2]))/n\_i)\*100  
 p\_L3 <- (sum(!is.na(df\_PoT[, 3]))/n\_i)\*100  
 p\_L4 <- (sum(!is.na(df\_PoT[, 4]))/n\_i)\*100  
 p\_L5 <- (sum(!is.na(df\_PoT[, 5]))/n\_i)\*100  
 p\_L6 <- (sum(!is.na(df\_PoT[, 6]))/n\_i)\*100  
 p\_noTrt <- (sum(!is.na(df\_PoT[, 7]))/n\_i)\*100  
   
 results <- list(m\_M = m\_M, m\_C = m\_C, m\_E = m\_E, m\_L = m\_L,   
 m\_ToT = m\_ToT, m\_TH = m\_TH, m\_AE = m\_AE, m\_C\_IC\_cl = m\_C\_IC\_cl,  
 m\_C\_IC\_a = m\_C\_IC\_a,T2D\_data = T2D\_data,T2D\_data\_temp = T2D\_data\_temp, m\_curTrt = m\_curTrt, m\_D = m\_D, m\_Age = m\_Age,  
 tc = tc, te = te, tLY = tLY, tLY\_undisc = tLY\_undisc,   
 tc\_hat = tc\_hat, te\_hat = te\_hat, tLY\_hat = tLY\_hat, tLY\_undisc\_hat = tLY\_undisc\_hat,   
 p\_L1 = p\_L1, p\_L2 = p\_L2, p\_L3 = p\_L3, p\_L4 = p\_L4, p\_L5 = p\_L5, p\_L6 = p\_L6, p\_noTrt = p\_noTrt,   
 t\_L1= t\_L1, t\_L2 = t\_L2, t\_L3 = t\_L3, t\_L4 = t\_L4, t\_L5 = t\_L5, t\_L6 = t\_L6, t\_noTrt = t\_noTrt)  
 }  
   
 if(PSA == T){  
 results <- list(m\_M = m\_M, m\_C = m\_C, m\_E = m\_E, tc = tc , te = te, tc\_hat = tc\_hat, te\_hat = te\_hat, tLY\_hat = tLY\_hat, tLY\_undisc\_hat = tLY\_undisc\_hat)  
 }  
   
 if(mainresults == T){  
 results <- list(tc\_hat = tc\_hat, te\_hat = te\_hat)  
 }  
   
 return(results) # return the results  
 }) # end of with(l\_params)  
   
} # end of the MicroSim function