# Appendix 5 – R code

##### Simulation os survival times

# install below if required

library(survival)

library(survSens)

# Functions ---------------------------------------------------------------

expit <- function(X){exp(X)/(1+exp(X))}

treatment\_assign <- function(p){

###

# p - vector of probabilities for Bernoulli rv

###

X <- c()

for (pi in 1:length(p)){

X <- c(X,rbinom(1,1,p[pi]))

}

return(X)

}

# Parameters & Input ------------------------------------------------------

set.seed(874251)

# Total number of patients

N <- 300

# Simulate 2 confounders and 1 unmeasured confounder

X1 <- rnorm(N)

X2 <- rbinom(N,1,0.45)

U <- rbinom(N,1,0.5)

# Treatment model coefficients

a0 <- -1

a1 <- 0.35

a2 <- -0.25

a3 <- 0.693147

# Outcome model coefficients

b0 <- -0.23

b1 <- -0.2

b2 <- 0.2

b3 <- 1.25276

gamma <- -0.59784

# Weibull parameters - baseline hazard

lambda <- 1 # scale

v <- 1.5 # shape

# Random uniform sample for PIF

u <- runif(N)

# Censoring

theta <- 4 # censoring parameter

cen\_times <- runif(N,0,theta) # simulated censoring times

# max follow up

maxfup <- 5

# Simulation --------------------------------------------------------------

# Treatment model (predictors)

treatmod <- a0 + a1\*X1 + a1\*X2 + a3\*U

p <- expit(treatmod) # treatment assignment probability

D <- treatment\_assign(p)

# check treatment assignment balance

table(D)

# Outcome model - linear predictor

linpred <-exp(b0 + gamma\*D + b1\*X1 + b2\*X2 + b3\*U)

# Simulate survival times

Times <- -((log(u)/(lambda\*linpred)))

Times <- Times\*\*(1/v)

# Apply independent censoring mechanism

status <- ifelse(cen\_times>Times, 1,0)

# Max follow-up

for (t in 1:N){

if (Times[t]>maxfup){

Times[t] <- maxfup

status[t] <- 0

}

}

table(status)/N

# Simulate PFS data -------------------------------------------------------

# Weibull parameters

lambda <- 3 # scale

v <- 2 # shape

# Empty dataframe

times\_PFS <- data.frame(Time=NULL, status=NULL)

# Pragmatic algorithm to simualate PFS times and status

for (p in 1:N){

t\_pfs <- rweibull(1,v,lambda) # sample PFS time

if (t\_pfs<simulated\_data$Times[p]){

times\_PFS <- rbind.data.frame(times\_PFS, c(t\_pfs, 1)) # progression occured before death event

} else if (t\_pfs>=simulated\_data$Times[p] & simulated\_data$status[p]==1){

times\_PFS <- rbind.data.frame(times\_PFS, c(simulated\_data$Times[p], 1)) # progression = death

}else if (t\_pfs>=simulated\_data$Times[p] & simulated\_data$status[p]==0){

times\_PFS <- rbind.data.frame(times\_PFS, c(simulated\_data$Times[p], 0)) # censored before progression could occur

}

}

# Rename columns

colnames(times\_PFS) <- c("Times", "status")

# Summary of events vs. censoring

table(times\_PFS$status)

# Create PFS dataframe

simulated\_data\_PFS <- data.frame(Times=times\_PFS$Times, status=times\_PFS$status, D=D)

# Analysis ----------------------------------------------------------------

os\_data <- read.csv("..../simulated\_OS\_data.csv")

set.seed(32087)

# Naive analysis ----------------------------------------------------------

# Cox model no adjustment for UC

res.cox <- coxph(Surv(Times, status) ~ D + X1 + X2, data = os\_data)

# Truth -------------------------------------------------------------------

res.cox\_T <- coxph(Surv(Times, status) ~ D + X1 + X2 + U, data = os\_data)

# PFS ---------------------------------------------------------------------

pfs\_data <- read.csv("..../simulated\_PFS\_data.csv")

res.cox.PFS <- coxph(Surv(Times, status) ~ D, data = pfs\_data)

# SA1 - Huang 2020 --------------------------------------------------------

# Parameters:

# zetaT = range of the coefficients of U in the response model

# zetaZ = range of the coefficients of U in the treatment model

# theta = marginal probability of U = 1

thetas <- c(0.4,0.45,0.5, 0.55,0.6)

zetaTs <- c(0.8,1,1.2,1.4,1.6)

zetaZs <- c(0.55,0.6,0.65,0.7,0.75)

# Empty array for results

Huang\_pe <- array(NA, dim = c(length(zetaZs), length(zetaTs), length(thetas)))

Huang\_se <- array(NA, dim = c(length(zetaZs), length(zetaTs), length(thetas)))

for (t in 1:length(thetas)){

# theta value

theta <- thetas[t]

for (i in 1:length(zetaZs)){

for (j in 1:length(zetaTs)){

Hmodel <- survSensitivity(os\_data$Times, os\_data$status, os\_data$D, cbind(os\_data$X1,os\_data$X2),

"EM\_reg", zetaT = zetaTs[j], zetaZ = zetaZs[i], theta = theta)

Huang\_pe[i,j,t] <- Hmodel$tau1

Huang\_se[i,j,t] <- Hmodel$tau1.se

}

}

}

# SA2 - Ding 2016 ---------------------------------------------------------

# Parameters:

# RR\_EU = relative risk between the exposure and unmeasured confounder

# HR\_UD = hazard ratio of the

# Function to calculate BF

BF <- function(RR\_EU, HR\_UD){

factor <- (RR\_EU\*HR\_UD)/(RR\_EU+HR\_UD-1)

return(factor)

}

# Array of assumptions of association

RR\_EUs <- c(1.6, 1.8, 2, 2.2, 2.4)

HR\_UDs <- c(3.2, 3.3, 3.4,3.5,3.6, 3.7)

# Naive unadjusted

obs <- summary(res.cox)$coef[1,2]

obsciL <-summary(res.cox)$conf.int[,"lower .95"][1]

obsciU <-summary(res.cox)$conf.int[,"upper .95"][1]

# Apply the transform for non rare outcomes and use the RR form

RR\_UDs <- (1-0.5\*\*sqrt(HR\_UDs))/(1-0.5\*\*sqrt(1/HR\_UDs))

# Empty matrix for results

ding\_mat\_pe <- matrix(nrow = length(RR\_EUs), ncol = length(RR\_UDs))

ding\_mat\_lci <- matrix(nrow = length(RR\_EUs), ncol = length(RR\_UDs))

ding\_mat\_uci <- matrix(nrow = length(RR\_EUs), ncol = length(RR\_UDs))

# Calculate outcome for combinations of assumptions

for (i in 1:length(RR\_EUs)){

for (j in 1:length(RR\_UDs)){

adjustment <- BF(RR\_EUs[i], RR\_UDs[j])

ding\_mat\_pe[i,j] <- obs/adjustment

ding\_mat\_lci[i,j] <- obsciL/adjustment

ding\_mat\_uci[i,j] <- obsciU/adjustment

}

}