Quick Guide

JIA, Ruofan

January 2024

1 MAIC

MAIC reweights the IPD in index study by the effect modifiers. After weighting, the moments of the effect modifiers in index study should be matched with those in competitor study. The weight w_i for each participant i in $j = 1$ is calculated by:

$$ln(w_i) = ln[\frac{Pr(j = 2|x_i)}{1 - Pr(j = 2|x_i)}] = \alpha_0 + \alpha_1 x_i$$

The estimation of α_1 is obtained by minimizing:

$$Q(\alpha_1) = \sum_{i=1}^{n} exp(x_i^* \alpha_1)$$

Therefore, the estimation of the weight is:

$$w_i^* = exp(x_i^* \hat{\alpha}_1)$$

MAIC for time-to-event data relies on Cox regression. After obtaining the weights, the hazard ratio between two treatments in competitor study is calculated by minimizing:

$$\sum_{i \quad \text{with event at} \quad t_i} log \frac{\hat{w}_i e^{x_i^T \beta + D \times 1(k=2)}}{\sum_{j \quad \text{at risk at} \quad t_i} \hat{w}_j e^{x_i^T \beta + D \times 1(k=2)}}$$

where D is the relative effects between treatment B and C.

The code for conducting MAIC can be found in MAIC.R.

2 NMA

This is a two-step method for NMA of survival analysis. In the first step, the parameters of an appropriate survival distributions are estimated for each treatment in each study, for example, μ_{kj} and η_{kj} for weibull distribution. For studies only with AgD, the IPD are reconstructed from the K-M curves. The estimates of the distribution-specific parameters are transformed into a normally distributed scale denoted by $\tilde{\mu}_{kj}$ and $\tilde{\eta}_{kj}$. The covariance of the estimates S_{kj} for each treatment in each study can also be obtained in step 1. In the second step, the transformed estimates of the parameters are synthesised. with a multivariate NMA model.

$$\begin{bmatrix} \tilde{\mu}_{kj} \\ \tilde{\eta}_{kj} \end{bmatrix} \sim N(\begin{bmatrix} \mu_{kj} \\ \eta_{kj} \end{bmatrix}, S_{jk})$$

where $\begin{bmatrix}\mu_{kj}\\ \eta_{kj}\end{bmatrix} = \begin{cases}\begin{bmatrix}\mu_{1j}\\ \eta_{1j}\end{bmatrix}, k=1\\ \begin{bmatrix}\mu_{1j}\\ \eta_{1j}\end{bmatrix} + \begin{bmatrix}\delta_{j1k\mu}\\ \delta_{j1k\eta}\end{bmatrix}, k>1\end{cases}$, and $\begin{bmatrix}\delta_{j1k\mu}\\ \delta_{j1k\eta}\end{bmatrix}$ can be fixed or random.
The code for conducting NMA can be found in NMA.R.

3 ML-NMR

3.1 Individual Data

We use the parametric Survival regression model to model the IPD.
Individual level:

$$P(t_{ikj}|x,\mu_j,\eta_j,\boldsymbol{\beta}_1,\boldsymbol{\beta}_{2,k},\gamma_k) = f_{kj}(t|x,\mu_j,\eta_j,\boldsymbol{\beta}_1,\boldsymbol{\beta}_{2,k},\gamma_k)$$
$$S_{kj}(t|x,\mu_j,\eta_j,\boldsymbol{\beta}_j,\boldsymbol{\beta}_{2,k},\gamma_k) = 1-\int_{-\infty}^{t} f_{kj}(y|x,\mu_j,\eta_j,\boldsymbol{\beta}_1,\boldsymbol{\beta}_{2,k},\gamma_k)dy$$
$$\lambda_{kj}(t|x,\mu_j,\eta_j,\boldsymbol{\beta}_j,\boldsymbol{\beta}_{2,k},\gamma_k) = \frac{f_{kj}(t|x)}{S_{kj}(t|x)}$$

where f_{kj} is an appropriate density for the survival data, S_{kj} is the survival function, and $\lambda_{kj}(t|x)$ is the hazard function.

3.2 Aggregate Data

From the KM curve which is reported in almost all of the survival analysis as shown in Figure 1, we can obtain some summary statistics $T_{\cdot kj}$ such as mean and median from the studies only involving AgD by reconstructing the event and censoring times[**?**]. We can use the software DigitizeIt (https://www.digitizeit.xyz/) to extract the x-axis and y-axis coordinates from the published KM curve. Suppose that distribution of $T_{\cdot kj}$ can be found and the parameter of the corresponding distribution is $\theta_{\cdot kj}$.
Aggregate level:

$$P(T_{\cdot kj}) \approx g(T_{\cdot kj};\theta_{\cdot kj})$$

3.3 Relation between IPD and AgD

In most of the cases, $\theta_{\cdot kj}$ is the expectation of some function w.r.t t_{ikj}, denoted by $U(t_{ikj}|\mu_j,\eta_j,\boldsymbol{\beta}_1,\boldsymbol{\beta}_{2,k},\gamma_k)$. Therefore, $\theta_{\cdot kj}$ can be expressed as a function of $\mu_j,\eta_j,\boldsymbol{\beta}_1,\boldsymbol{\beta}_{2,k},\gamma_k$.
Relation:

$$E(U(t_{ikj}|x,\mu_j,\eta_j,\boldsymbol{\beta}_1,\boldsymbol{\beta}_{2,k},\gamma_k)) = \int U(t|x,\mu_j,\eta_j,\boldsymbol{\beta}_1,\boldsymbol{\beta}_{2,k},\gamma_k) f_{kj}(t|x,\mu_j,\eta_j,\boldsymbol{\beta}_1,\boldsymbol{\beta}_{2,k},\gamma_k)dt$$
$$\theta_{\cdot kj} = \int E(U(t_{ikj}|x,\mu_j,\eta_j,\boldsymbol{\beta}_1,\boldsymbol{\beta}_{2,k},\gamma_k)) f_x(x)dx$$
$$\approx \sum_{\tilde{N}_{kj}} E(U(t_{ikj}|\tilde{x},\mu_j,\eta_j,\boldsymbol{\beta}_1,\boldsymbol{\beta}_{2,k},\gamma_k))$$

Since the integral $\int E(U(t_{ikj}|x,\mu_j,\eta_j,\boldsymbol{\beta}_1,\boldsymbol{\beta}_{2,k},\gamma_k)) f_x(x)dx$ is hard to compute, we can use the numerical method. We can generate random samples from $f(x)$ which is joint distribution of the covariates. The information of $f(x)$ is usually not included in the studies with only AgD. But the summary statistics of x are often available as shown in Figure 2. Therefore, with the assumption that the correlation among covariates are constant across trials, we can use the Gaussian copula to generate $\tilde{x}$ and use Monte-Carlo method to approximate this integral. Let $\tilde{N_{kj}}$ be the number of simulated samples.

There are two functions that can be used to fit different models. The functions are stored in mlnmr.R and mlnmrtrans.R. See the Table 1 for more details.

Table 1: Functions for ML-NMR

Distribution for IPD	Summary Statistics for AgD	Function
Weibull	Mean	mlnmr
Weibull	Median	mlnmr
Gompertz	Mean	mlnmr
Log Normal	Mean	mlnmrtrans
Log Logistic	Median	mlnmrtrans

3.4 Details of the functions

3.4.1 mlnmr

Input:

```
#ns:number of studies

#n_cov: number of covariates

#nt: number of treatments

#n_list: list of sample size.
#n_list[[i]][[j]] contains the sample size of treatment j in study i.

#t_list: list of t data.
#For study i with IPD, t_list[[i]][[j]] contains the vector of t of treatment j;
#For study i with AgD, t_list[[i]][[j]] contains the summary statistics
#of treatment j in study i

#X_list: list of covariates.
#For study i with IPD, X_list[[i]][[j]] contains the design matrix of treatment j;
#For study i with AgD, X_list[[i]][[j]] contains the simulated X of treatment j
#in study i

#delta_list: list of the truncation information of studies with IPD.
#delta_list[[i]][[j]] contains the truncation information of treatment j
#in study i

#n_simu_list: list of number of simulated sample size. Only for study with AgD.
#n_simu_list[[i]][[j]] contains the simulated sample size for treatment j
#in study i.

#var_list: for gompertz only; the estimated variance of t for AgD.
#var_list[[i]][[j]] contains the variance of t of treatment j in study i

#tr_list: list of the treatments in each study;
#tr_list[[i]] contains the vector of treatments in study i

#st_list: list of the studies containing treatment;
#st_list[[i]] contains the vector of studies containing treatment i

#sd_pr: sd in the proposal normal distribution
```

```
#n_ipd:number of studies with IPD

#n_agd:number of studies with AgD

#type:weibull_median;weibull_mean;gompertz

#burnin: burn-in interations for Gibbs sampling

#nloop: number of posterior samples

#mu_init: vector of initial values of mu of length ns

#eta_init: vector of initial values of eta of length ns

#gamma_init: vector of initial values of gamma of length nt
#with first value being 0

#beta1_init: vector of initial values of beta1 of length n_cov

#beta2_init: vector of initial values of beta2 of length n_cov

#Note: the baseline treatment should be treatment 1;
#the treatments should be placed in order for tr_list;
#The study with IPD should be labelled first
```

Output: A list of posterior mean and samples. The first five elements in the list contain the posterior mean of μ, η, γ, β_1 and β_2. The 6-10 elements contains the corresponding posterior samples.

3.4.2 mlnmrtrans

Input:

```
#ns:number of studies

#n_cov: number of covariates

#nt: number of treatments

#n_list: list of sample size.
#For IPD, n_list[[i]][[j]][[1]] contains the noncensored sample
#size of treatment j in study i;n_list[[i]][[j]][[2]] contains
#the censored sample size of treatment j in study i;For AgD,
#n_list[[i]][[j]] contains the sample size of treatment j in
#study i

#t_list: list of transformed t data (log t).
#For IPD, t_list[[i]][[j]][[1]] contains the vector of
#noncensored t of treatment j in study i;t_list[[i]][[j]][[2]]
#contains the vector of censored t of treatment j in study
#i;For AgD, t_list[[i]][[j]] contains the summary statistics of
#t of treatment j in study i

#X_list: list of covariates.
#For IPD, X_list[[i]][[j]][[1]] contains the  design matrix of
#noncensored sample of treatment j in study i;X_list[[i]][[j]]
```

```
#[[2]] contains the design matrix of censored sample of
#treatment j in study i;For AgD, X_list[[i]][[j]] contains the
#simulated X of treatment j in study i

#n_simu_list: list of number of simulated sample size. Only for study with AgD.
#n_simu_list[[i]][[j]] contains the simulated sample size for treatment j
#in study i.

#var_list: for gompertz only; the estimated variance of t for AgD.
#var_list[[i]][[j]] contains the variance of t of treatment j in study i

#tr_list: list of the treatments in each study;
#tr_list[[i]] contains the vector of treatments in study i

#st_list: list of the studies containing treatment;
#st_list[[i]] contains the vector of studies containing treatment i

#sd_pr: sd in the proposal normal distribution

#n_ipd:number of studies with IPD

#n_agd:number of studies with AgD

#type:logn;loglog

#burnin: burn-in interations for Gibbs sampling

#nloop: number of posterior samples

#mu_init: vector of initial values of mu of length ns

#eta_init: vector of initial values of eta of length ns

#gamma_init: vector of initial values of gamma of length nt
#with first value being 0

#beta1_init: vector of initial values of beta1 of length n_cov

#beta2_init: vector of initial values of beta2 of length n_cov

#Note: the baseline treatment should be treatment 1;
#the treatments should be placed in order for tr_list;
#The study with IPD should be labelled first
```

Output: A list of posterior mean and samples. The first five elements in the list contain the posterior mean of μ, η, γ, β_1 and β_2. The 6-10 elements contains the corresponding posterior samples.

4 Example

4.1 Data Generating Process

There are two studies, of which study 1 with IPD compares between treatment A and B, and study 2 with AgD compares between treatment A and C. The sample size of each study is 200, and the allocation ratio is 1:1. There are 5 covariates, of which the first three come from multivariate normal distribution, the fourth comes from gamma distribution and the fifth comes from binomial distribution. Suppose t follows

weibull distribution.

```
library(MASS)
library(survival)

ns<-2 #Number of studies
nt<-3 #Number of treatments
n_cov<-5 #Number of covariates

n_AB<-200 #sample size of each study
n_AC<-200

beta1<-c(0.1,0.1,-0.1,0.1,-0.1)
beta2<-c(-0.2,0.2,-0.2,-0.2,0.2)
gammaB<--1.2
gammaC<--0.5
eta1<-0.8
eta2<-1.2
mu1<-6.2
mu2<-5.8
shape1<-eta1
shape2<-eta2

#Sample size in each treatment
n1_A<-n_AB/2;n1_B<-n_AB/2;n2_A<-n_AC/2;n2_C<-n_AC/2;


#Generate the covariates
sigma=diag(1,5)
sigma[1,2]<-sigma[2,1]<-sigma[3,2]<-sigma[2,3]<-0.5
sigma[1,3]<-0.1
x_AB<-cbind(mvrnorm(n_AB,mu=rep(0,3),sigma),rgamma(n_AB,4,2),rbinom(n_AB,1,0.2))
sigma=diag(1,5)
sigma[1,2]<-sigma[2,1]<-sigma[3,2]<-sigma[2,3]<-0.5
sigma[1,3]<-0.1
x_AC<-cbind(mvrnorm(n_AC,mu=rep(0.5,3),sigma),rgamma(n_AC,6,2),rbinom(n_AC,1,0.7))

scale1<-1/c(mu1^(1/eta1)*exp((x_AB[1:(n_AB/2),]%*%(beta1))/eta1),mu1^(1/eta1)*exp(
scale2<-1/c(mu2^(1/eta2)*exp((x_AC[1:(n_AC/2),]%*%(beta1))/eta2),mu2^(1/eta2)*exp(


#Generate t
t1<-c()
for (i in 1:n_AB){

     t1[i]<-rweibull(1,shape=shape1,scale=scale1[i])
}


t2<-c()
for (i in 1:n_AC){

      t2[i]<-rweibull(1,shape=shape2,scale=scale2[i])
}
```

```
#Censoring
t1_censored<-c()
delta1<-c()

for (i in 1:n_AB){
      if (t1[i]>0.6){
        t1_censored[i]<-0.6
        delta1[i]<-0
      }else{
        t1_censored[i]<-t1[i]
        delta1[i]<-1
      }
}


t2_censored<-c()
delta2<-c()

for (i in 1:n_AC){
      if (t2[i]>0.45){
        t2_censored[i]<-0.45
        delta2[i]<-0
      }else{
        t2_censored[i]<-t2[i]
        delta2[i]<-1
      }
    }

#transform to data frame
x_AB<-as.data.frame(x_AB)
colnames(x_AB)<-c('x1','x2','x3','x4','x5')
x_AC<-as.data.frame(x_AC)
colnames(x_AC)<-c('x1','x2','x3','x4','x5')
```

4.2 MAIC

To conduct MAIC, we first need to load the MAIC package.

```
#install.packages('maic')
library(maic)
```

Then we calculate the weights. Note we need the summary statistics of covariates in study 2. The details can be found in https://www.rdocumentation.org/packages/maic/versions/0.1.4/topics/maicWeight

```
data_AB<-cbind(rep(0,n1_A),rep(0,n1_B),x_AB)
colnames(data_AB)<-c('tr',colnames(x_AB))

target <- c("x1_mean" = mean(x_AC[,1]),
                  "x2_mean" = mean(x_AC[,2]),
                  "x3_mean" = mean(x_AC[,3]),
                  "x4_mean" = mean(x_AC[,4]),
                  "x5_mean" = mean(x_AC[,5]))

dict <- data.frame(
      "match.id" =
        c("x1", "x2",
          "x3", "x4", "x5"),
```

```
      "target.variable" =
        c("x1_mean", "x2_mean","x3_mean","x4_mean","x5_mean"),
      "index.variable" =
        c("x1", "x2",
          "x3",
          "x4",
          "x5"),
      "match.type" =
        c("mean", "mean", "mean", "mean", "mean"),
      stringsAsFactors = FALSE)

ipmat <- createMAICInput(
      index = x_AB,
      target = target,
      dictionary = dict,
      matching.variables =
        c("x1", "x2",
          "x3", "x4", "x5"))

wts <- maicWeight(ipmat)
```

At last we compare treatment A and B in study 2.

```
x_AB<-as.matrix(x_AB)
fre_res1<-coxph(Surv(t1_censored,delta1)~x_AB*c(rep(0,n_AB/2),rep(1,n_AB/2)),weight
```

```
> fre_res1
Call:
coxph(formula = Surv(t1_censored, delta1) ~ x_AB * c(rep(0, n_AB/2),
    rep(1, n_AB/2)), weights = wts)

                                                coef exp(coef) se(coef) robust se
z      p
x_ABx1                                       0.75679   2.13143  0.40830   0.31669
2.390 0.0169
x_ABx2                                       0.23115   1.26005  0.29675   0.21159
1.092 0.2746
x_ABx3                                       0.47152   1.60243  0.34166   0.29598
1.593 0.1111
x_ABx4                                       0.29005   1.33650  0.23612   0.18347
1.581 0.1139
x_ABx5                                       0.71759   2.04949  0.49666   0.41514
1.729 0.0839
c(rep(0, n_AB/2), rep(1, n_AB/2))           -2.12925   0.11893  2.12443   1.67082 -1.1
x_ABx1:c(rep(0, n_AB/2), rep(1, n_AB/2)) -0.94222   0.38976  0.59893   0.42906 -2.1
x_ABx2:c(rep(0, n_AB/2), rep(1, n_AB/2))  0.44891   1.56660  0.51985   0.47248
0.950 0.3421
x_ABx3:c(rep(0, n_AB/2), rep(1, n_AB/2)) -0.33515   0.71523  0.57538   0.52563 -0.6
x_ABx4:c(rep(0, n_AB/2), rep(1, n_AB/2))  0.17430   1.19041  0.49442   0.39234
0.444 0.6569
x_ABx5:c(rep(0, n_AB/2), rep(1, n_AB/2))  0.06532   1.06750  1.04807   0.93009
0.070 0.9440

Likelihood ratio test=31.48  on 11 df, p=0.0009234
n= 200, number of events= 157
```

We usually can get the hazard ratio estimate of studies with AgD. In our case the hazard ratio of treatment

C and B in study 2 is:

```
#Note this is usually reported in AgD
x_AC<-as.matrix(x_AC)
fre_res2<-coxph(Surv(t2_censored,delta2)~x_AC*c(rep(0,n_AC/2),rep(1,n_AC/2)))

#Tha hazard ratio of C and B:
exp(fre_res2$coefficients[6]-fre_res1$coefficients[6])
```

4.3 NMA

To conduct NMA, we first load the required packages. Note that to use survivalnma, winbugs should be installed.

```
#install.packages('survminer')
#install.packages('survivalnma')
library(survminer)
library(survivalnma)
```

The input is the estimated K-M curve. Here we directly calculate the K-M curve based on IPD. For AgD with K-M curves, we can extract the needed points from the reported K-M curves. Section 5 will introduce how to do this.

```
fit <- survfit(Surv(t1_censored[1:(n_AB/2)],delta1[1:(n_AB/2)]) ~ 1)

 d <- data.frame(time = fit$time,
                 n.risk = fit$n.risk,
                 n.event = fit$n.event
 )

 write.table(d,'D://survival//NMA//simu//samplesize//ln//s6//n400//s1t1.txt')

 fit <- survfit(Surv(t1_censored[(n_AB/2+1):n_AB],delta1[(n_AB/2+1):n_AB]) ~ 1)

 d <- data.frame(time = fit$time,
                 n.risk = fit$n.risk,
                 n.event = fit$n.event
 )

 write.table(d,'D://survival//NMA//simu//samplesize//ln//s6//n400//s1t2.txt')




 fit <- survfit(Surv(t2_censored[1:(n_AC/2)],delta2[1:(n_AC/2)]) ~ 1)

 d <- data.frame(time = fit$time,
                 n.risk = fit$n.risk,
                 n.event = fit$n.event
 )

 write.table(d,'D://survival//NMA//simu//samplesize//ln//s6//n400//s2t1.txt')

 fit <- survfit(Surv(t2_censored[(n_AC/2+1):n_AC],delta2[(n_AC/2+1):n_AC]) ~ 1)

 d <- data.frame(time = fit$time,
                 n.risk = fit$n.risk,
```

```
                n.event = fit$n.event
  )

  write.table(d,'D://survival//NMA//simu//samplesize//ln//s6//n400//s2t3.txt')
```

Then we can use the NMA function to calculate the adjusted parameters. More details of survnma can be found in https://github.com/certara/survivalnma. Note sometimes the model can not converge.

```
nma_df <- data.frame(
    stringsAsFactors = FALSE,
    "treatment" = c("A", "B","A",  "C"),
    "study" = c("Study 1", "Study 1", "Study 2", "Study 2"),
    "baseline" = c("A", "A", "A", "A"),
    "filepath" = c("D://survival//NMA//simu//samplesize//ln//s6//n400//s1t1.txt",
                   "D://survival//NMA//simu//samplesize//ln//s6//n400//s1t2.txt",
                   "D://survival//NMA//simu//samplesize//ln//s6//n400//s2t1.txt",
                   "D://survival//NMA//simu//samplesize//ln//s6//n400//s2t3.txt"))

fit_wbl <- survnma(nma_df, model="weibull",bugs.directory =

"D://winbugs//WinBUGS14",min_time_change = 0.000001)

#fit_wbl$fit$mean stores the estimates of parameters;
#fit_wbl$fit$sims.matrix stores the posterior samples
```

4.4 ML-NMR

4.4.1 Weibull

This section introduces how to construct the input if we want to use weibull distribution to model the IPD.

Firstly we write a function for generating the simulated X of AgD. Note that this function is written for this special example and can not be used in other situations.

```
cov_simu5<-function(x_dat,cor_refer,n_simu){
  n_cov<-dim(x_dat)[2]
  x1_mean<-mean(x_dat[,1])
  x1_var<-var(x_dat[,1])
  x2_mean<-mean(x_dat[,2])
  x2_var<-var(x_dat[,2])
  x3_mean<-mean(x_dat[,3])
  x3_var<-var(x_dat[,3])

  x4_mean<-mean(x_dat[,4])
  x4_var<-var(x_dat[,4])

  x5_p<-x5_mean<-mean(x_dat[,5])

  x4_scale<-x4_var/x4_mean
  x4_shape<-x4_mean/x4_scale

  ## Initialization and parameters
  P <- cor_refer
  d <- nrow(P)                                  # Dimension
  n_simu <- n_simu                                      # Number of samples
```

```
  ## Simulation (non-vectorized version)
  A <- t(chol(P))
  U <- matrix(nrow = n_simu, ncol = d)
  for (i in 1:n_simu){
    Z      <- rnorm(d)
    X      <- A%*%Z
    U[i, ] <- pnorm(X)
  }

  ## Simulation (compact vectorized version)
  U <- pnorm(matrix(rnorm(n_simu*d), ncol = d) %*% chol(P))

  X_simu<-matrix(0,nrow=n_simu,ncol=n_cov)

  for (i in 1:n_simu){
    X_simu[i,1]<-qnorm(U[i,1],x1_mean,sqrt(x1_var))
    X_simu[i,2]<-qnorm(U[i,2],x2_mean,sqrt(x2_var))
    X_simu[i,3]<-qnorm(U[i,3],x3_mean,sqrt(x3_var))
    X_simu[i,4]<-qgamma(U[i,4],x4_shape,scale=x4_scale)
    X_simu[i,5]<-ifelse(U[i,5]<x5_p,1,0)
  }
  return(X_simu)
}
```

Then we construct the input of the functions.

```
ns=2
n_cov=5
nt=3
n_ipd=1
n_agd=1
type='weibull_median'

n_simu=10000 #sample size of simulated X
burnin<-20000
nloop<-10000

x1_A<-x_AB[1:(n_AB/2),]
x1_B<-x_AB[(n_AB/2+1):n_AB,]

x2_A<-x_AC[1:(n_AC/2),]
x2_C<-x_AC[(n_AC/2+1):n_AC,]

t1_A<-t1_censored[1:n1_A]
t1_B<-t1_censored[(n1_A+1):(n1_A+n1_B)]

delta1_A<-delta1[1:n1_A]
delta1_B<-delta1[(n1_A+1):(n1_A+n1_B)]

#Simulate X for AgD
n2_simu_A<-n_simu;n2_simu_C<-n_simu;
X2_simu_A<-cov_simu5(x2_A,cor(x_AB),n2_simu_A);X2_simu_C<-cov_simu5(x2_C,cor(x_AB)

 n_list<-list()
  n_list[[1]]<-list()
  n_list[[1]][[1]]<-n1_A
```

```
n_list[[1]][[2]]<-n1_B
n_list[[2]]<-list()
n_list[[2]][[1]]<-n2_A
n_list[[2]][[3]]<-n2_C


X_list<-list()
X_list[[1]]<-list()
X_list[[1]][[1]]<-x1_A
X_list[[1]][[2]]<-x1_B
X_list[[2]]<-list()
X_list[[2]][[1]]<-X2_simu_A
X_list[[2]][[3]]<-X2_simu_C

n_simu_list<-list()
n_simu_list[[2]]<-list()
n_simu_list[[2]][[1]]<-n2_simu_A
n_simu_list[[2]][[3]]<-n2_simu_C


t_list<-list()
t_list[[1]]<-list()
t_list[[2]]<-list()
t_list[[1]][[1]]<-t1_A
t_list[[1]][[2]]<-t1_B
t_list[[2]][[1]]<-mean(t2[1:n2_A])
t_list[[2]][[3]]<-mean(t2[(n2_A+1):(n2_A+n2_C)])


delta_list<-list()
delta_list[[1]]<-list()
delta_list[[1]][[1]]<-delta1_A
delta_list[[1]][[2]]<-delta1_B

tr_list<-list()
tr_list[[1]]<-c(1,2)
tr_list[[2]]<-c(1,3)

st_list<-list()
st_list[[1]]<-c(1,2,3,4)
st_list[[2]]<-c(1)
st_list[[3]]<-c(2)
```

4.4.2 Weibull

This section introduces how to construct the input if we want to use log normal distribution to model the IPD.

We construct the input of the functions.

```
ns=2
n_cov=5
nt=3
n_ipd=1
n_agd=1
```

```
type='logn'

n_simu=10000 #sample size of simulated X
burnin<-20000
nloop<-10000

#take log
t1_censored<-log(t1_censored)

#separate censored samples and non-censored samples
t1_A_ncensored<-t1_censored[1:(n_AB/2)][which(delta1[1:(n_AB/2)]==1)]
t1_A_censored<-t1_censored[1:(n_AB/2)][which(delta1[1:(n_AB/2)]==0)]

t1_B_ncensored<-t1_censored[(n_AB/2+1):n_AB][which(delta1[(n_AB/2+1):n_AB]==1)]
t1_B_censored<-t1_censored[(n_AB/2+1):n_AB][which(delta1[(n_AB/2+1):n_AB]==0)]

x1_A_ncen<-x_AB[which(delta1[1:(n_AB/2)]==1),]
x1_A_cen<-x_AB[which(delta1[1:(n_AB/2)]==0),]
x1_B_ncen<-x_AB[(n_AB/2+1):n_AB,][which(delta1[(n_AB/2+1):n_AB]==1),]
x1_B_cen<-x_AB[(n_AB/2+1):n_AB,][which(delta1[(n_AB/2+1):n_AB]==0),]

x2_A<-x_AC[1:(n_AC/2),]
x2_C<-x_AC[(n_AC/2+1):n_AC,]

n1_A_ncen<-length(t1_A_ncensored);n1_A_cen<-length(t1_A_censored)
n1_B_ncen<-length(t1_B_ncensored);n1_B_cen<-length(t1_B_censored)


#Simulate X for AgD
n2_simu_A<-n_simu;n2_simu_C<-n_simu;
X2_simu_A<-cov_simu5(x2_A,cor(x_AB),n2_simu_A);X2_simu_C<-cov_simu5(x2_C,cor(x_AB)

n_list<-list()
n_list[[1]]<-list()
n_list[[1]][[1]]<-list()
n_list[[1]][[2]]<-list()
n_list[[1]][[1]][[1]]<-n1_A_ncen
n_list[[1]][[1]][[2]]<-n1_A_cen
n_list[[1]][[2]][[1]]<-n1_B_ncen
n_list[[1]][[2]][[2]]<-n1_B_cen
n_list[[2]]<-list()
n_list[[2]][[1]]<-n2_A
n_list[[2]][[3]]<-n2_C



X_list<-list()
X_list[[1]]<-list()
X_list[[1]][[1]]<-list()
X_list[[1]][[2]]<-list()
X_list[[1]][[1]][[1]]<-x1_A_ncen
X_list[[1]][[1]][[2]]<-x1_A_cen
X_list[[1]][[2]][[1]]<-x1_B_ncen
X_list[[1]][[2]][[2]]<-x1_B_cen
X_list[[2]]<-list()
X_list[[2]][[1]]<-X2_simu_A
```

```
X_list[[2]][[3]]<-X2_simu_C


n_simu_list<-list()
n_simu_list[[2]]<-list()
n_simu_list[[2]][[1]]<-n2_simu_A
n_simu_list[[2]][[3]]<-n2_simu_C



t_list<-list()
t_list[[1]]<-list()
t_list[[2]]<-list()
t_list[[1]][[1]]<-list()
t_list[[1]][[2]]<-list()
t_list[[1]][[1]][[1]]<-t1_A_ncensored
t_list[[1]][[1]][[2]]<-t1_A_censored
t_list[[1]][[2]][[1]]<-t1_B_ncensored
t_list[[1]][[2]][[2]]<-t1_B_censored
t_list[[2]][[1]]<-mean(log(t2[1:n2_A]))
t_list[[2]][[3]]<-mean(log(t2[(n2_A+1):(n2_A+n2_C)]))


  tr_list<-list()
  tr_list[[1]]<-c(1,2)
  tr_list[[2]]<-c(1,3)

  st_list<-list()
  st_list[[1]]<-c(1,2,3,4)
  st_list[[2]]<-c(1)
  st_list[[3]]<-c(2)
```

5 Extract K-M Curves

The code in this section can be found in KM.RWe first get a K-M curve reported in the paper. Then we load the required packages:

```
library(IPDfromKM)

library(haven)
library(survival)
library(sas7bdat)
library(dplyr)
library(data.table)
library(ggplot2)
library(ggfortify)
library(survival)
library(survminer)
library(ggsurvfit)
library(gtsummary)
library(tidycmprsk)
library(condSURV)
```

We get the points of the K-M curves by clicking the points on the curves following the instructions.

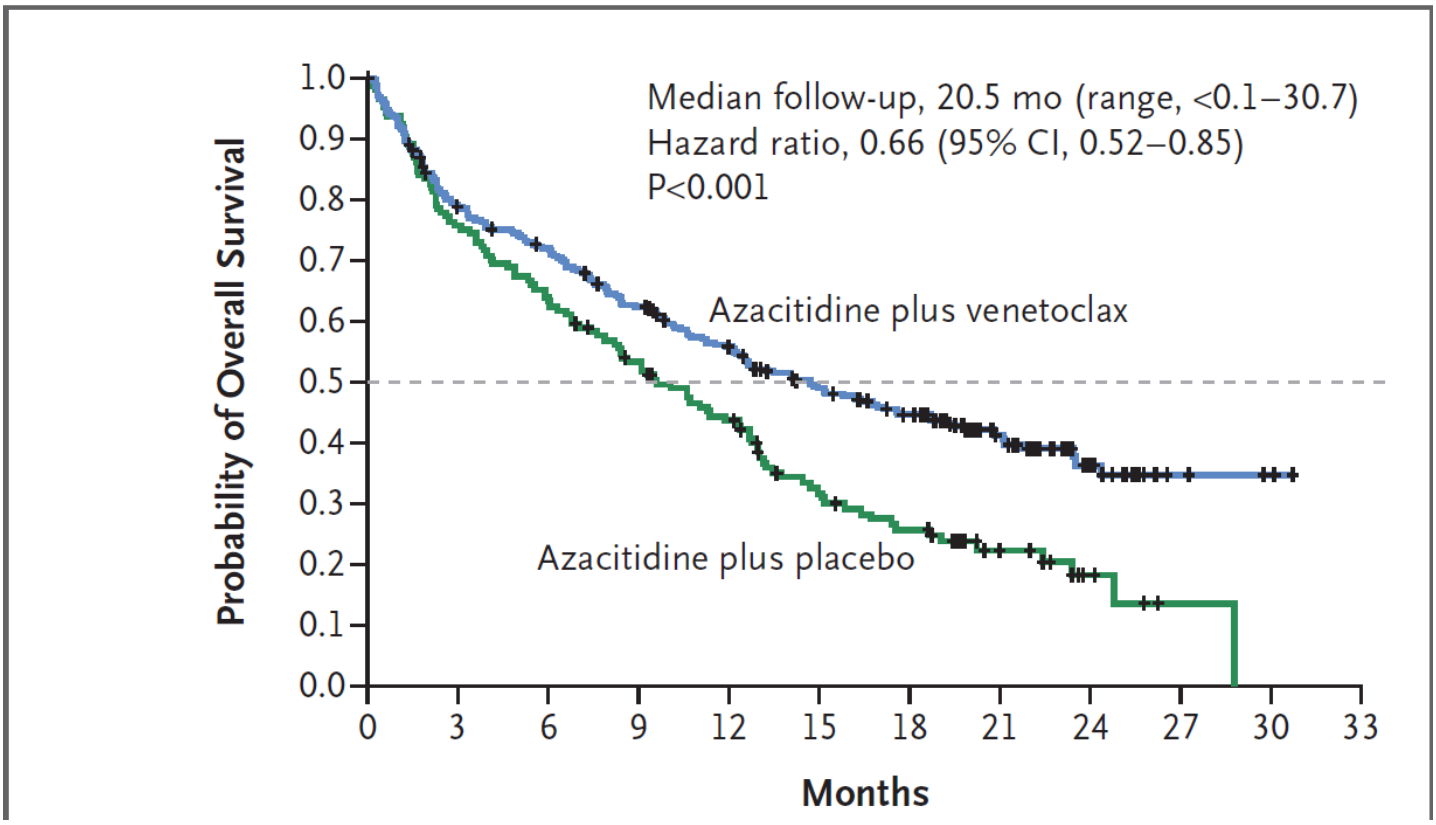

Figure 1: K-M curve (Dinardo)

```
pts <- getpoints("C://study//survival//KM//Dinardo.PNG", 0, 33, 0, 1)
```

Input the time points and samples at risks.

```
trisk <- c(0, 3, 6, 9, 12, 15, 18, 21, 24, 27, 30,33)
nrisk.ven <- c(145, 109, 92, 74, 59, 38, 30, 14, 5, 1,0, 0)
```

Get the IPD.

```
pre_ven <- preprocess(dat=pts, trisk=trisk, nrisk=nrisk.ven,totalpts=NULL,maxy=1)
data_ven <- getIPD(prep=pre_ven, armID=1, tot.events=NULL)
ipd_ven <- data_ven$IPD
```

We can get the summary statistics from the IPD.

```
mean(ipd_ven$time)
median(ipd_ven$time)
sd(ipd_ven$time)
mean(log(ipd_ven$time))
median(log(ipd_ven$time))
sd(log(ipd_ven$time))
```