Sensitivity Analysis - Ovarian

Florian Stijven

29-3-2022

```
## Warning: package 'flexsurv' was built under R version 4.0.5

## Loading required package: survival

## Warning: package 'survival' was built under R version 4.0.5

## Warning: package 'Surrogate' was built under R version 4.0.5

## Warning: package 'tidyverse' was built under R version 4.0.5

## -- Attaching packages --------------------------------------- tidyverse 1.3.1 --

## v ggplot2 3.3.3     v dplyr   1.0.6
## v tibble  3.1.1     v stringr 1.4.0
## v tidyr   1.1.3     v forcats 0.5.1
## v readr   1.4.0

## Warning: package 'tibble' was built under R version 4.0.5

## Warning: package 'tidyr' was built under R version 4.0.5

## -- Conflicts ------------------------------------------ tidyverse_conflicts() --
## x dplyr::filter() masks stats::filter()
## x dplyr::lag()    masks stats::lag()
```

```
#put data in correct format
data = Ovarian %>% select(Pfs, Surv, Treat, PfsInd, SurvInd)
data = data %>% filter(Pfs <= Surv)
table(data$PfsInd, data$SurvInd)
```

```
##
##       0   1
##   0 215   0
##   1  26 950
```

```
data %>% filter(Surv == Pfs & PfsInd == 1 & SurvInd == 1) %>% nrow()
```

```
## [1] 200
```

Introduction

The `Ovarian` data set contains the progression-free survival times (PFS), and overall survival times (OS) for 1192 ovarian cancer patients. One observation is left out because the PFS is longer than OS, which is not possible. Both PFS and OS can be censored. Due to the nature of the endpoints, PFS is always smaller than or equal to OS. For 200 out of the 1191 observations, the observed times for PFS and OS are equal. With the copula based approach, this atomicity is however ignored. Moreover, the copula based approach does not impose ordering restrictions. Depending on the data, this could lead to issues.

Also note that due to the nature of the endpoints, PFS and OS are by definition associated. Indeed, PFS is a composite endpoint combining time-to-progression and time-to-death, whichever comes first.

Several conventional parametric models were considered for modelling the marginal survival functions: Weibull, log-logistic and generalized gamma. However, these all fail to describe the data well. This is caused by the fact that in the data, the hazard is initially very high, but later becomes very small. This is seen on the marginal survival functions were intially there is a large drop, but after some time the survival functions become almost constant at about 0.1-0.2. Therefore, a Royston-Parmar spline model is fitted using a two-stage approach.

Model Fitting

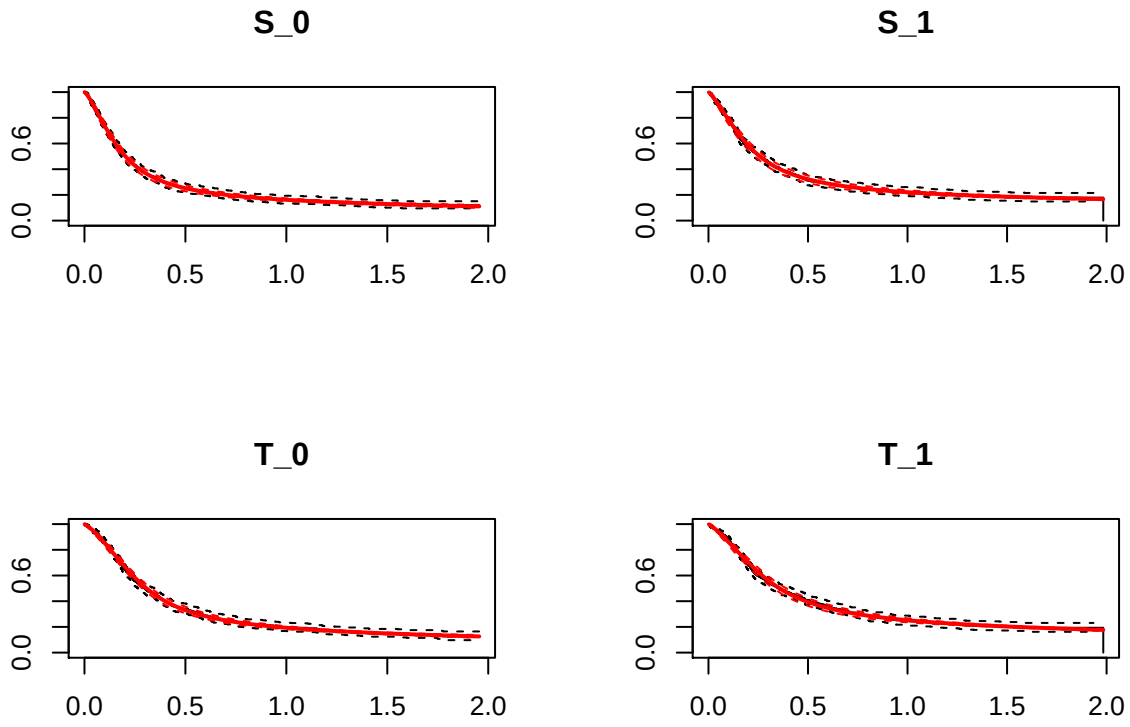
Royston-Parmar Spline Model

The Royston-Parmar spline model is fitted for each potential outcome *separately*, so four times. A simplification would be fit it once for PFS and once for OS, and then include a treatment effect into the model. This however entails and additional PH-assumption which is not needed anywhere in the further approach.

All Royston-Parmar spline models are fitted with two knots using the `flexsurvspline` function from the `flexsurv` package. So each survival function is modelled by 4 parameters. As can be seen in the plots below, the KM-estimate and parametric estimate of the respective survival functions are very near, indicating that this parametric model is appropriate.

```
new_data = data
par(mfrow = c(2,2))
fit_s0 = flexsurvspline(formula = Surv(Pfs, PfsInd)~1, data = data,
                        subset = data$Treat == 0, k = 2, scale = "hazard")
plot(fit_s0, main = "S_0")
new_data[new_data$Treat == 0, 1] = 1 - predict(fit_s0, type = "survival",
                                               newdata = data[1,],
                                               times = data[data$Treat == 0, 1])$.pred[[1]][,2]
fit_s1 = flexsurvspline(formula = Surv(Pfs, PfsInd)~1, data = data,
                        subset = data$Treat == 1, k = 2, scale = "hazard")
new_data[new_data$Treat == 1, 1] = 1 - predict(fit_s1, type = "survival",
                                               newdata = data[4,],
                                               times = data[data$Treat == 1, 1])$.pred[[1]][,2]
plot(fit_s1, main = "S_1")
fit_t0 = flexsurvspline(formula = Surv(Surv, SurvInd)~1, data = data,
                        subset = data$Treat == 0, k = 2, scale = "hazard")
new_data[new_data$Treat == 0, 2] = 1 - predict(fit_t0, type = "survival",
                                               newdata = data[1,],
                                               times = data[data$Treat == 0, 2])$.pred[[1]][,2]
plot(fit_t0, main = "T_0")
fit_t1 = flexsurvspline(formula = Surv(Surv, SurvInd)~1, data = data,
```

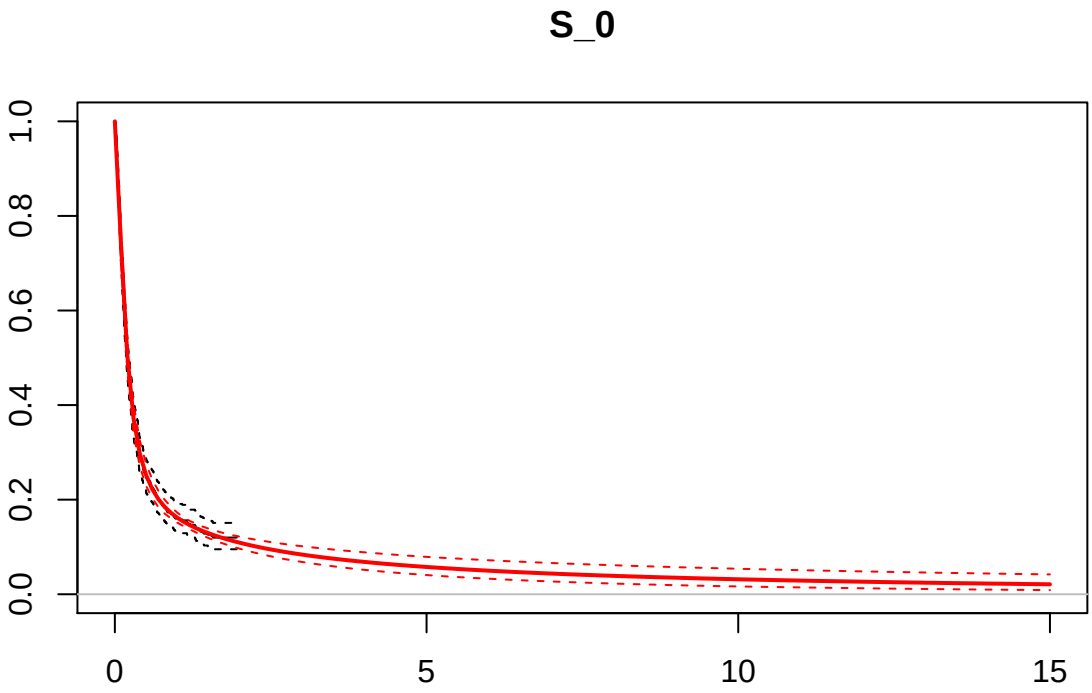
```
                          subset = data$Treat == 1, k = 2, scale = "hazard")
new_data[new_data$Treat == 1, 2] = 1 - predict(fit_t1, type = "survival",
                                               newdata = data[4,],
                                               times = data[data$Treat == 1, 2])$.pred[[1]][,2]
plot(fit_t1, main = "T_1")
```



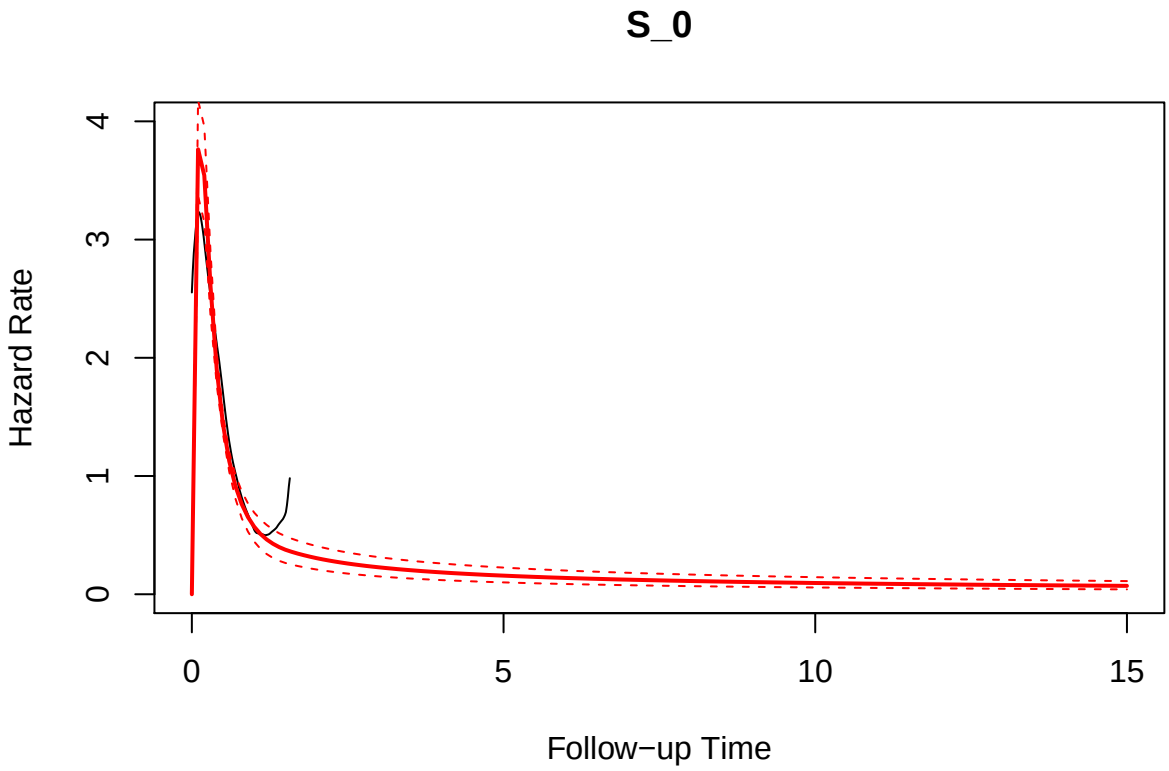
One particular issue with these data is that the maximum follow-up time is 2 years while a considerable proportion of patients survive past that point. The fitted parametric model however implies a survival and hazard function beyond two years. The survival and hazard function beyond those two years are however unobservable.

This is shown for S_0. The modelled survival function beyond the KM-estimate is implied by the model, but this is not verifiable. Still, this unverifiable part influences the metrics of surrogacy. However, it is not immediately clear how large this influence is.

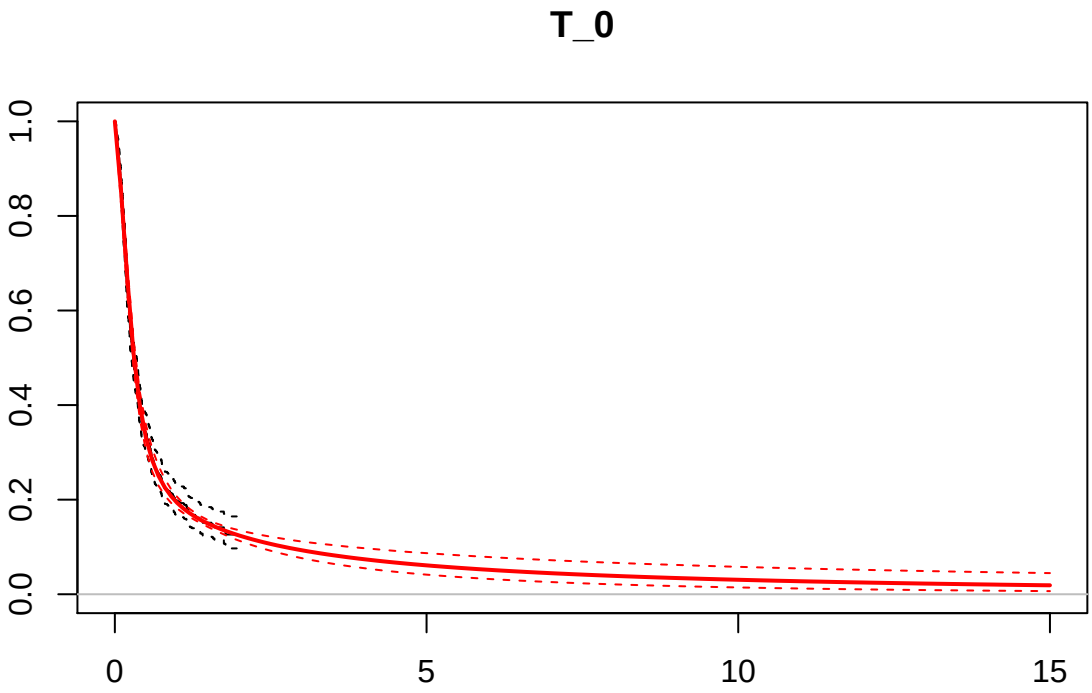
```
plot(fit_s0, main = "S_0", type = "survival", t = seq(0, 15, 0.1),
     xlim = c(0, 15))
abline(a = 0, b = 0, col = "gray")
```



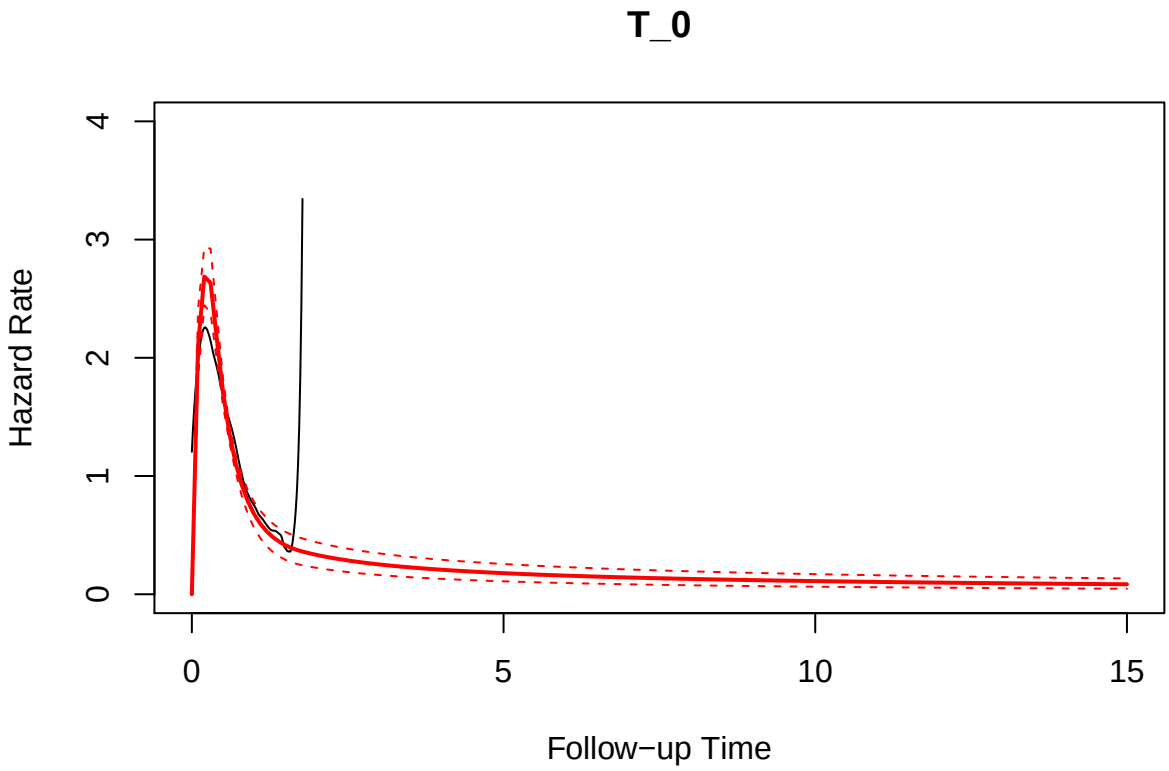

```
plot(fit_s0, main = "S_0", type = "hazard", t = seq(0, 15, 0.1),
     xlim = c(0, 15), ylim  =c(0, 4))
```



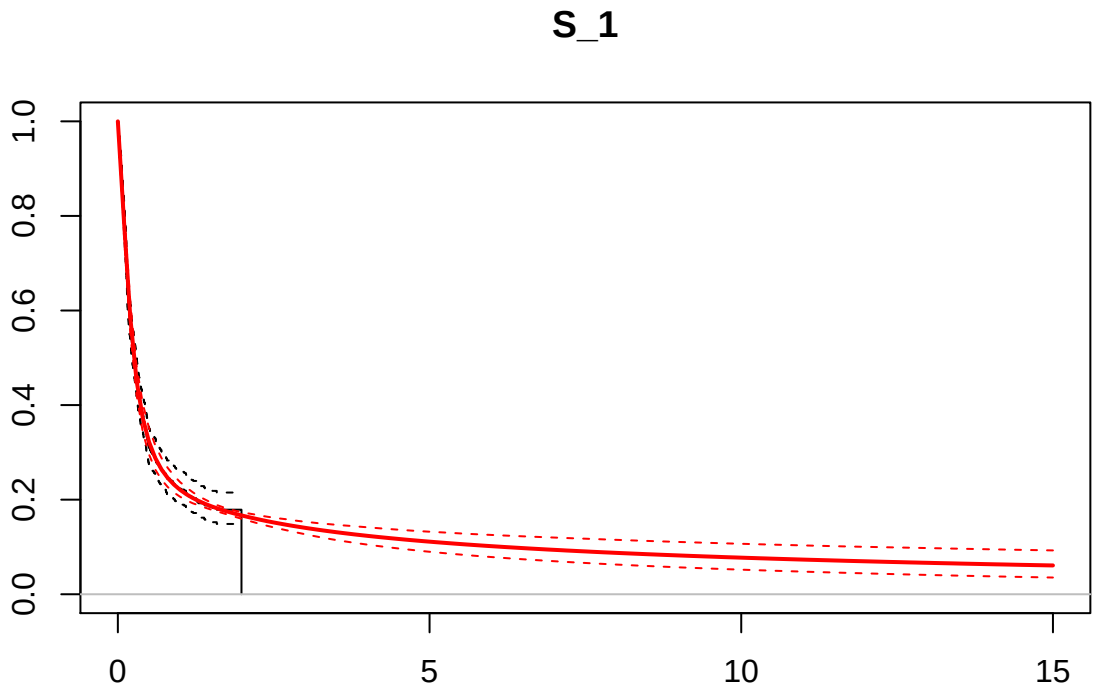

```
plot(fit_t0, main = "T_0", type = "survival", t = seq(0, 15, 0.1),
     xlim = c(0, 15))
abline(a = 0, b = 0, col = "gray")
```



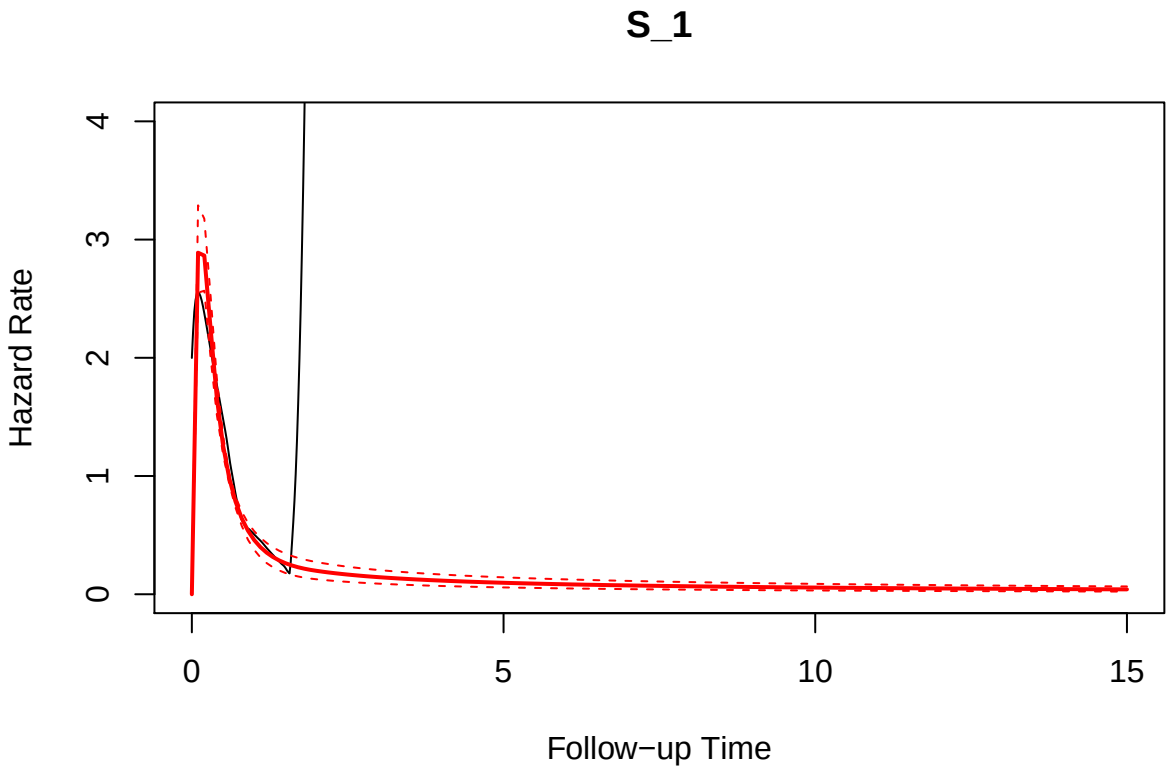

```
plot(fit_t0, main = "T_0", type = "hazard", t = seq(0, 15, 0.1),
     xlim = c(0, 15), ylim  =c(0, 4))
```



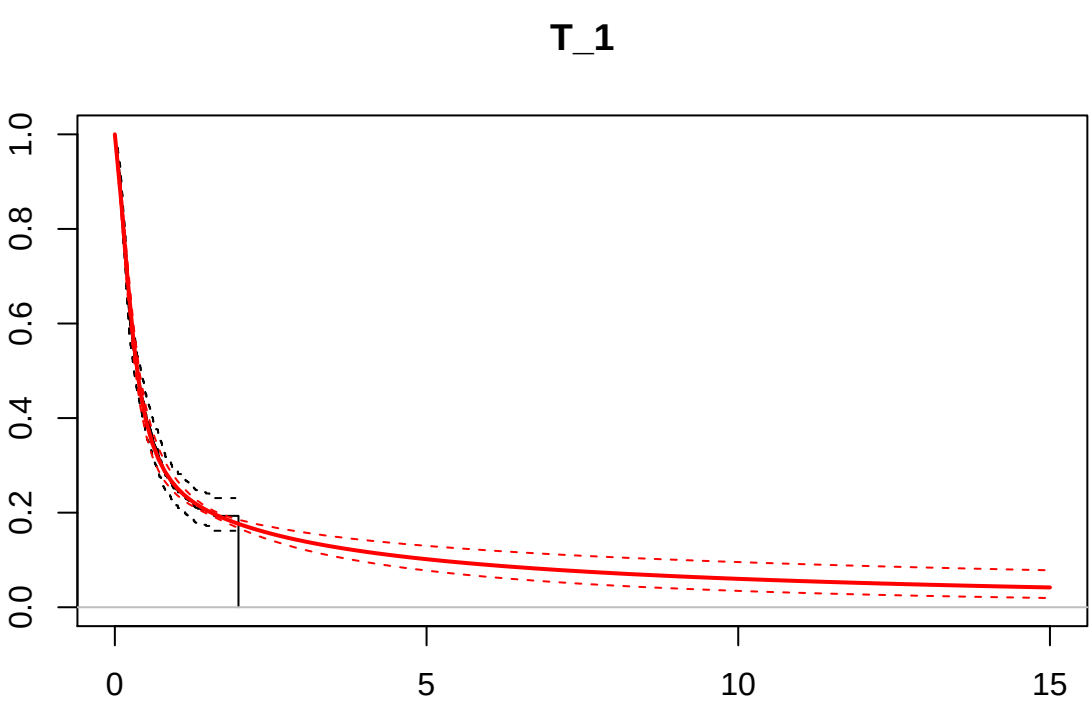

```
plot(fit_s1, main = "S_1", type = "survival", t = seq(0, 15, 0.1),
     xlim = c(0, 15))
abline(a = 0, b = 0, col = "gray")
```



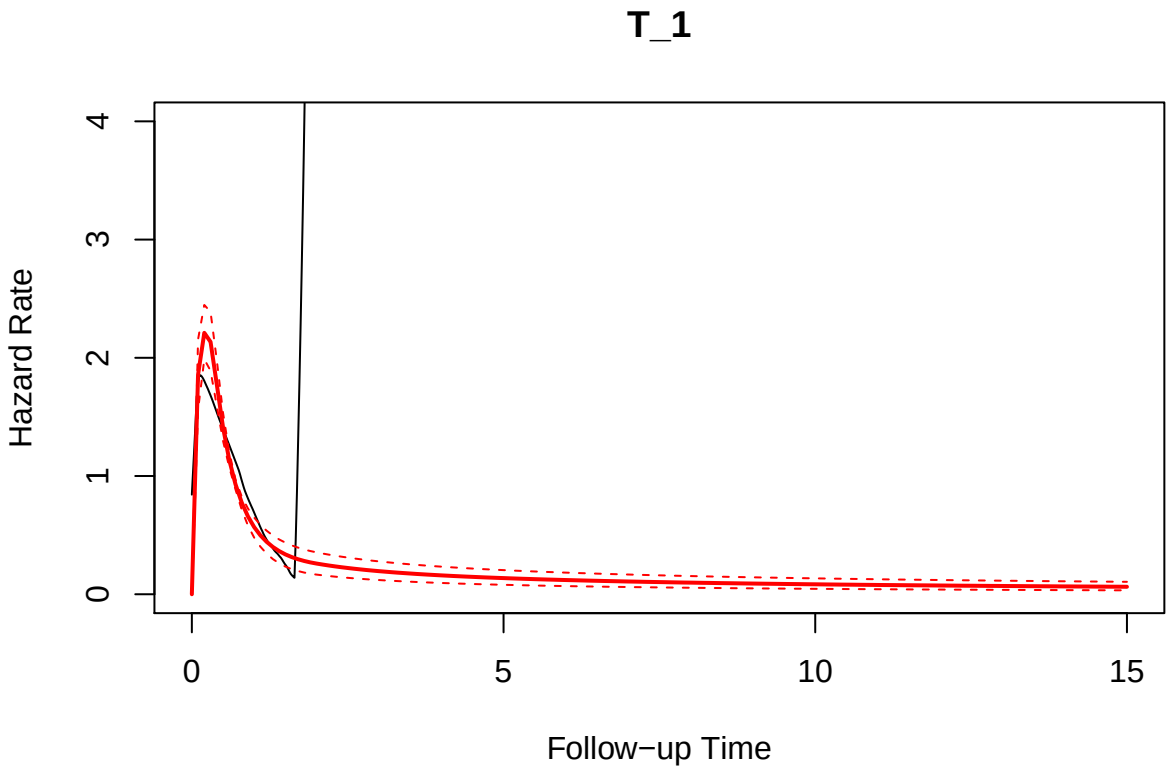

```
plot(fit_s1, main = "S_1", type = "hazard", t = seq(0, 15, 0.1),
     xlim = c(0, 15), ylim  =c(0, 4))
```




```
plot(fit_t1, main = "T_1", type = "survival", t = seq(0, 15, 0.1),
     xlim = c(0, 15))
abline(a = 0, b = 0, col = "gray")
```




```
plot(fit_t1, main = "T_1", type = "hazard", t = seq(0, 15, 0.1),
     xlim = c(0, 15), ylim  =c(0, 4))
```




```
predict(fit_s0, newdata = data[1,], type = "survival", times = c(15, 30, 60))$.pred
```

```
## [[1]]
##   .time       .pred
## 1    15 0.021032419
## 2    30 0.009328934
## 3    60 0.003486992
```

```
predict(fit_t0, newdata = data[1,], type = "survival", times = c(15, 30, 60))$.pred
```

```
## [[1]]
##   .time       .pred
## 1    15 0.018904513
## 2    30 0.007084533
## 3    60 0.002082726
```

```
predict(fit_s1, newdata = data[1,], type = "survival", times = c(15, 30, 60))$.pred
```

```
## [[1]]
##   .time      .pred
## 1    15 0.06086680
## 2    30 0.03833019
## 3    60 0.02236246
```

```
predict(fit_t1, newdata = data[1,], type = "survival", times = c(15, 30, 60))$.pred
```

```
## [[1]]
##   .time      .pred
## 1    15 0.04181985
## 2    30 0.02015913
## 3    60 0.00821696
```

Copula Model

The copula model is fitted in the second stage. To this end, the orginal PFS and OS times are transformed to uniform variables using the fitted survival functions. These uniform variables are then used to fit the copula model. Two parameters are fitted with this copula model: ρ_{S_0,T_0} and ρ_{S_1,T_1}. The other four correlation parameters are part of the sensitivity analysis.

```
cop_fit_ctrl = fit_copula(data = new_data[new_data$Treat == 0,], inits = 2.5)
```

```
## iter    2 value -204.304062
## iter    4 value -204.316913
## final  value -204.316913
## converged
```

```
cop_fit_trt = fit_copula(data = new_data[new_data$Treat == 1,], inits = 2.5)
```

```
## iter    2 value -228.440508
## iter    4 value -231.806467
## iter    6 value -231.806832
## final  value -231.806832
## converged
```

```
rho13 = (exp(cop_fit_ctrl$par[1]) - 1)/(exp(cop_fit_ctrl$par[1]) + 1)
rho24 = (exp(cop_fit_trt$par[1]) - 1)/(exp(cop_fit_trt$par[1]) + 1)
```

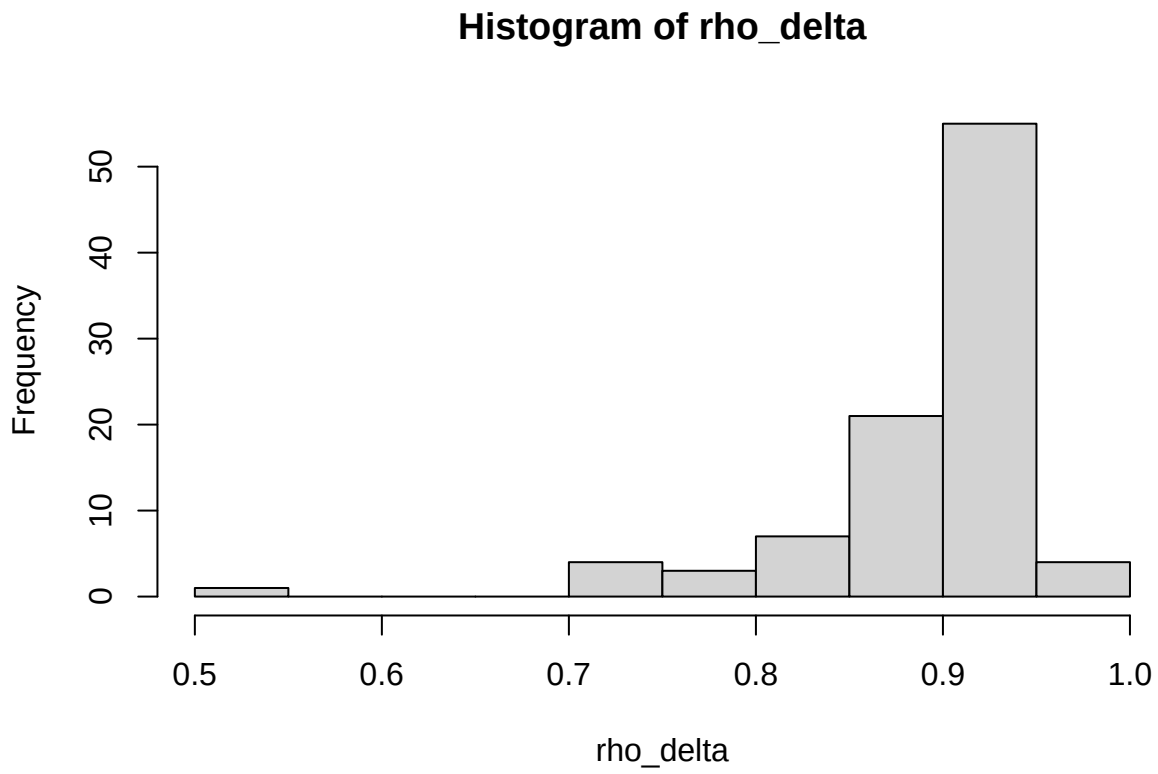
The observable correlations are 0.8851453 for the control group, and 0.9199036 for the treated group.

Sensitivity Analysis

The sensitivity analysis consists of sampling positive definite matrices. Given such a matrix, the measures of surrogacy can be derived. In this case, only ρ_Δ is considered as a measure of surrogacy because it easily follows from the normal copula parameters.

```
grid = seq(-1, 1, 0.2)
Pos.Def.Matrices(T0S0 = rho13, T1S1 = rho24,
                 T0T1 = grid, S0S1 = grid, T1S0 = grid, T0S1 = grid)
# Generated.Matrices = readRDS(file = "many_matrices.RData")
posdef_gen_matrices = Generated.Matrices[Generated.Matrices$Pos.Def.Status == 1,]
# saveRDS(object = Generated.Matrices, file = "many_matrices.RData")
rho_delta = numeric()
```

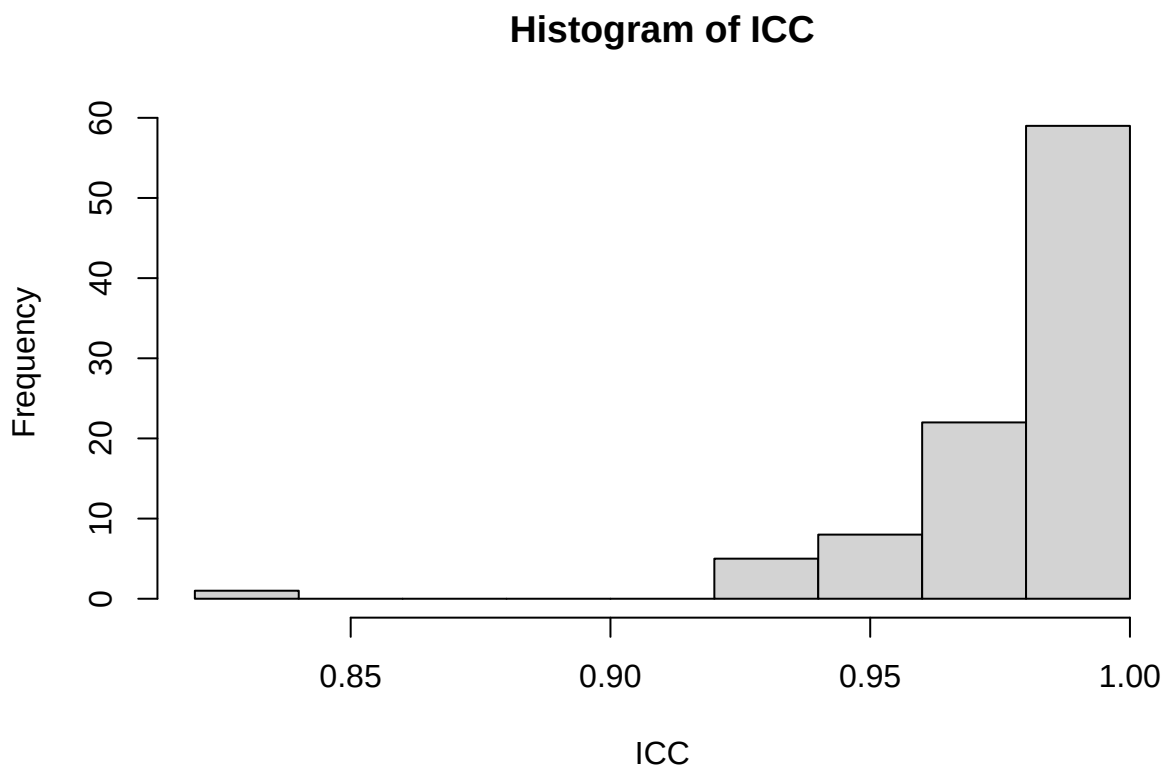
```
for(i in 1:nrow(posdef_gen_matrices)){
  num = posdef_gen_matrices[i, "T0S0"] + posdef_gen_matrices[i, "T1S1"] - posdef_gen_matrices[i, "T1S0"]
  denom = 2*sqrt((1 - posdef_gen_matrices[i, "T0T1"])*
                   (1 - posdef_gen_matrices[i, "S0S1"]))
  rho_delta = c(rho_delta, num/denom)
}
hist(rho_delta)
```



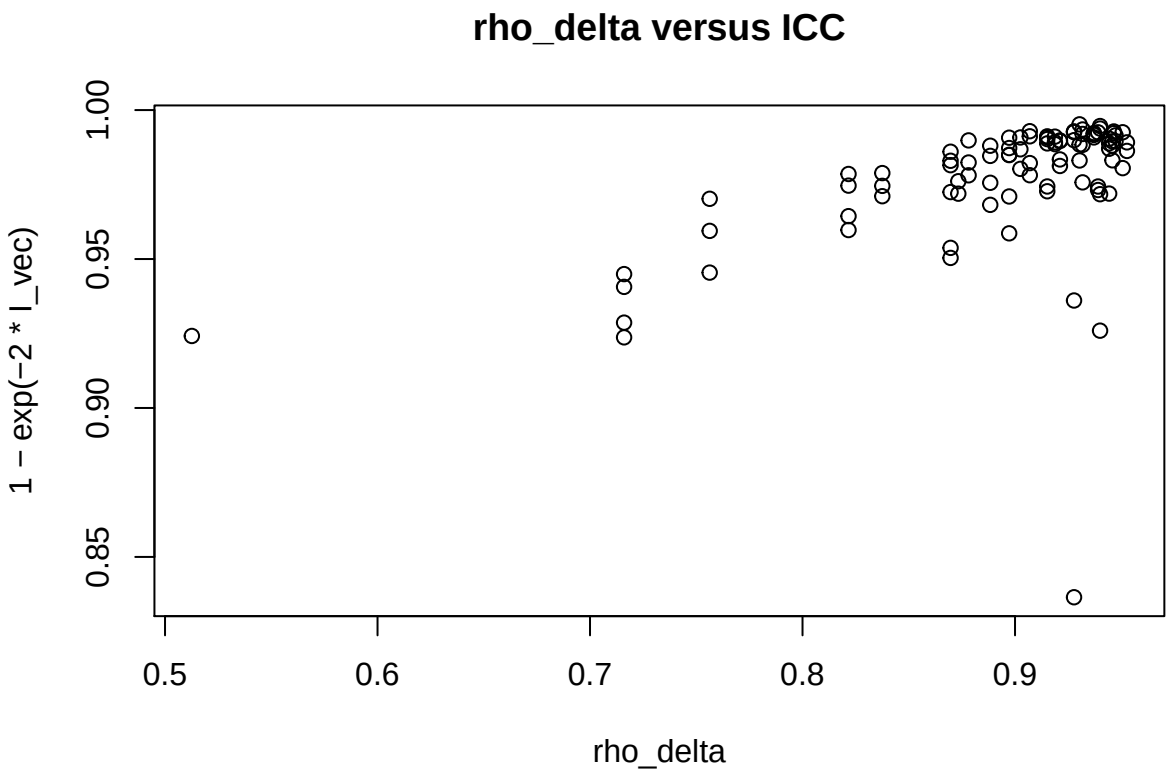

```
posdef_Sigma_list = as.list(1:nrow(posdef_gen_matrices))
for(i in 1:nrow(posdef_gen_matrices)){
  rho12 = posdef_gen_matrices[i, "S0S1"]
  rho13 = posdef_gen_matrices[i, "T0S0"]
  rho14 = posdef_gen_matrices[i, "T1S0"]
  rho23 = posdef_gen_matrices[i, "T0S1"]
  rho24 = posdef_gen_matrices[i, "T1S1"]
  rho34 = posdef_gen_matrices[i, "T0T1"]
  Sigma_temp = matrix(c(1, rho12, rho13, rho14,
                        rho12, 1, rho23, rho24,
                        rho13, rho23, 1, rho34,
                        rho14, rho24, rho34, 1), nrow = 4)
  posdef_Sigma_list[[i]] = Sigma_temp
}
library(parallel)
source("information_theoretic_functions.R")
cl = makeCluster(5)
```

```
clusterExport(cl, ls())
clusterEvalQ(cl, library(mvtnorm))
clusterEvalQ(cl, library(purrr))
clusterEvalQ(cl, library(flexsurv))
clusterEvalQ(cl, source("information_theoretic_functions.R"))
I_vec = unlist(parLapply(cl = cl, fun = r_h, X = posdef_Sigma_list,
                         n = 1000))
stopCluster(cl)
```

```
I_vec = readRDS(file = "I_vec.RData")
hist(1 - exp(-2*I_vec), main = "Histogram of ICC",
     xlab = "ICC")
```




```
plot(rho_delta, 1 - exp(-2*I_vec), main = "rho_delta versus ICC")
```

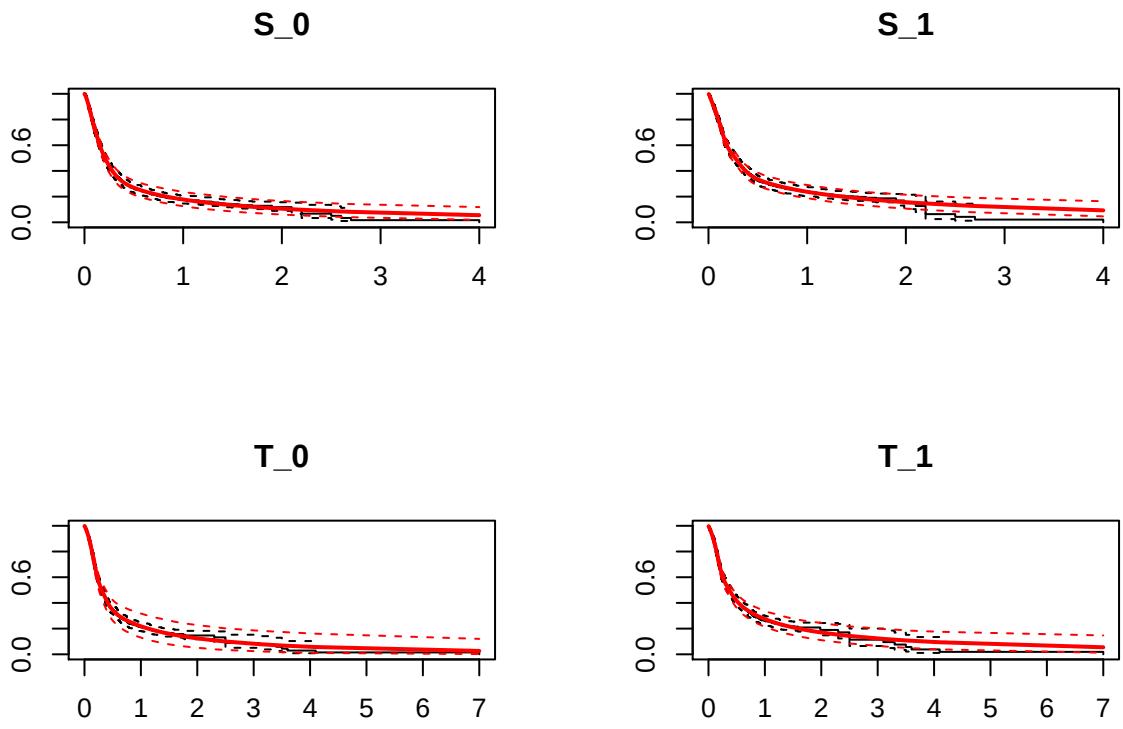


The histogram clearly indicates that across almost all realities compatible with the data, the ICA is very large. This indicates that PFS is a good surrogate for OS for the types of treatments and types of patients included in the data set. The histogram is based on 2085 positive definite matrices.

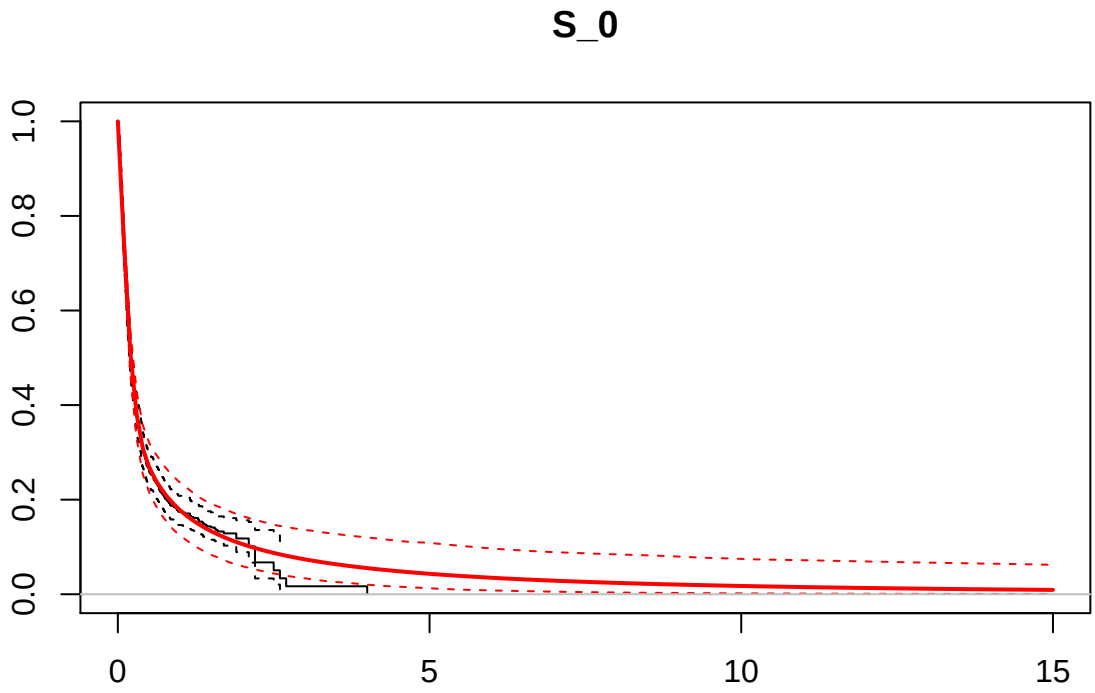
Sensitivity with respect to number of knots

```
#tweak the data a little by adding a death and pfs time after two years
data = rbind(data, c(2.7, 3.6, 0, 1, 1))
data = rbind(data, c(2.6, 3.4, 0, 1, 1))
data = rbind(data, c(2.5, 3.5, 0, 1, 1))
data = rbind(data, c(2.1, 3.1, 0, 1, 1))
data = rbind(data, c(2.2, 2.5, 0, 1, 1))
data = rbind(data, c(2.2, 2.5, 0, 1, 1))
data = rbind(data, c(1.7, 2.5, 0, 1, 1))
data = rbind(data, c(1.6, 2.3, 0, 1, 1))
data = rbind(data, c(1.9, 4.1, 0, 1, 1))
data = rbind(data, c(4, 7, 0, 1, 1))

data = rbind(data, c(2.7, 3.6, 1, 1, 1))
data = rbind(data, c(2.2, 3.3, 1, 1, 1))
data = rbind(data, c(2.5, 3.5, 1, 1, 1))
data = rbind(data, c(2.1, 3.1, 1, 1, 1))
data = rbind(data, c(2.2, 2.5, 1, 1, 1))
data = rbind(data, c(2.2, 2.5, 1, 1, 1))
```

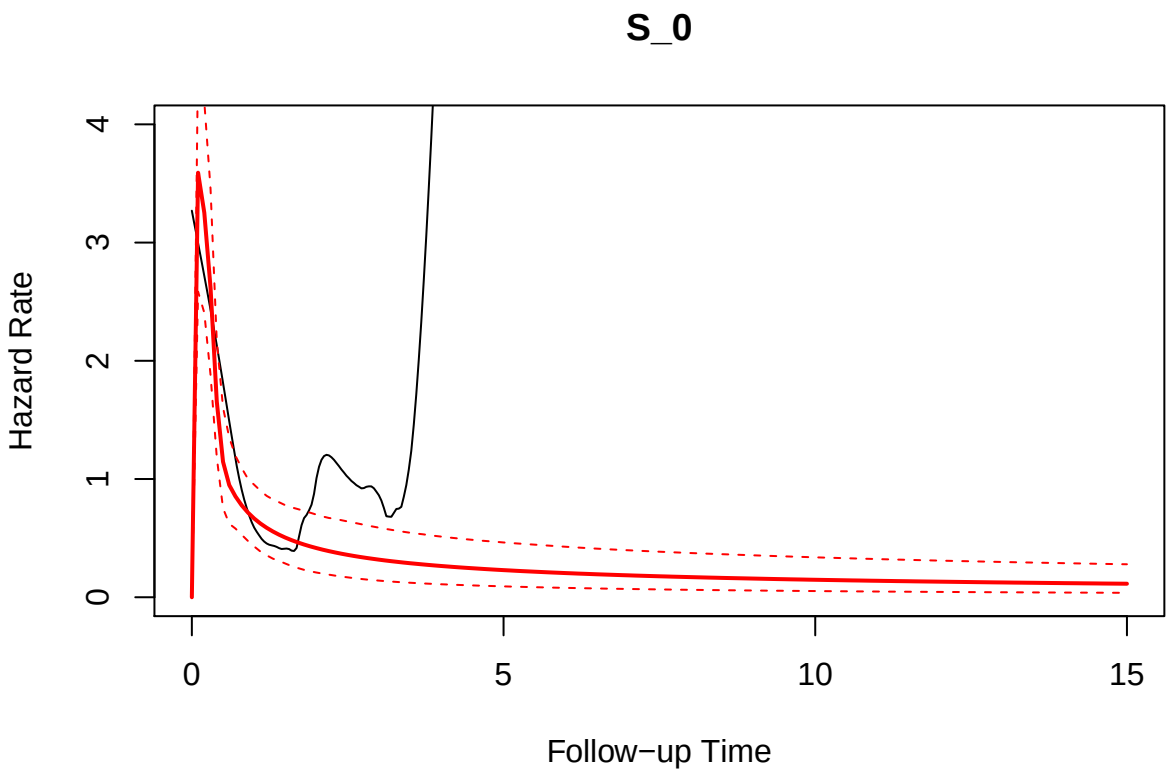
```
data = rbind(data, c(1.7, 2.5, 1, 1, 1))
data = rbind(data, c(1.6, 2.3, 1, 1, 1))
data = rbind(data, c(1.9, 4.1, 1, 1, 1))
data = rbind(data, c(4, 7, 1, 1, 1))

new_data = data
par(mfrow = c(2,2))
fit_s0 = flexsurvspline(formula = Surv(Pfs, PfsInd)~1, data = data,
                        subset = data$Treat == 0, k = 7, scale = "hazard")
plot(fit_s0, main = "S_0")
new_data[new_data$Treat == 0, 1] = 1 - predict(fit_s0, type = "survival",
                                               newdata = data[1,],
                                               times = data[data$Treat == 0, 1])$.pred[[1]][,2]
fit_s1 = flexsurvspline(formula = Surv(Pfs, PfsInd)~1, data = data,
                        subset = data$Treat == 1, k = 7, scale = "hazard")
new_data[new_data$Treat == 1, 1] = 1 - predict(fit_s1, type = "survival",
                                               newdata = data[4,],
                                               times = data[data$Treat == 1, 1])$.pred[[1]][,2]
plot(fit_s1, main = "S_1")
fit_t0 = flexsurvspline(formula = Surv(Surv, SurvInd)~1, data = data,
                        subset = data$Treat == 0, k = 7, scale = "hazard")
new_data[new_data$Treat == 0, 2] = 1 - predict(fit_t0, type = "survival",
                                               newdata = data[1,],
                                               times = data[data$Treat == 0, 2])$.pred[[1]][,2]
plot(fit_t0, main = "T_0")
fit_t1 = flexsurvspline(formula = Surv(Surv, SurvInd)~1, data = data,
                        subset = data$Treat == 1, k = 7, scale = "hazard")
new_data[new_data$Treat == 1, 2] = 1 - predict(fit_t1, type = "survival",
                                               newdata = data[4,],
                                               times = data[data$Treat == 1, 2])$.pred[[1]][,2]
plot(fit_t1, main = "T_1")
```



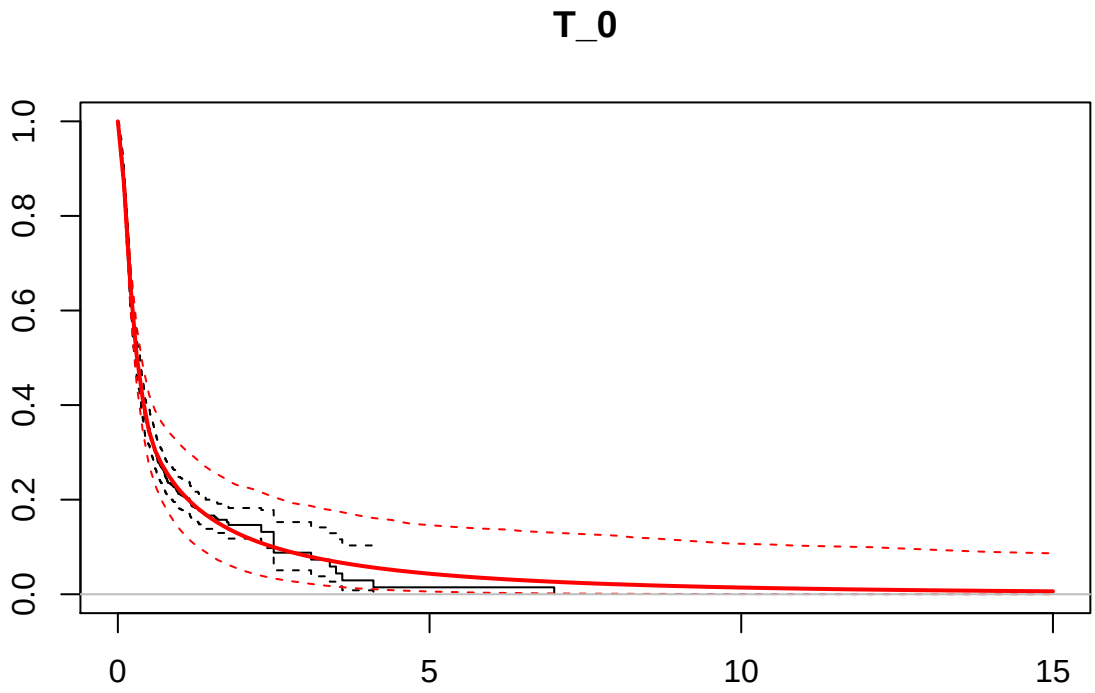

```
plot(fit_s0, main = "S_0", type = "survival", t = seq(0, 15, 0.1),
     xlim = c(0, 15))
abline(a = 0, b = 0, col = "gray")
```



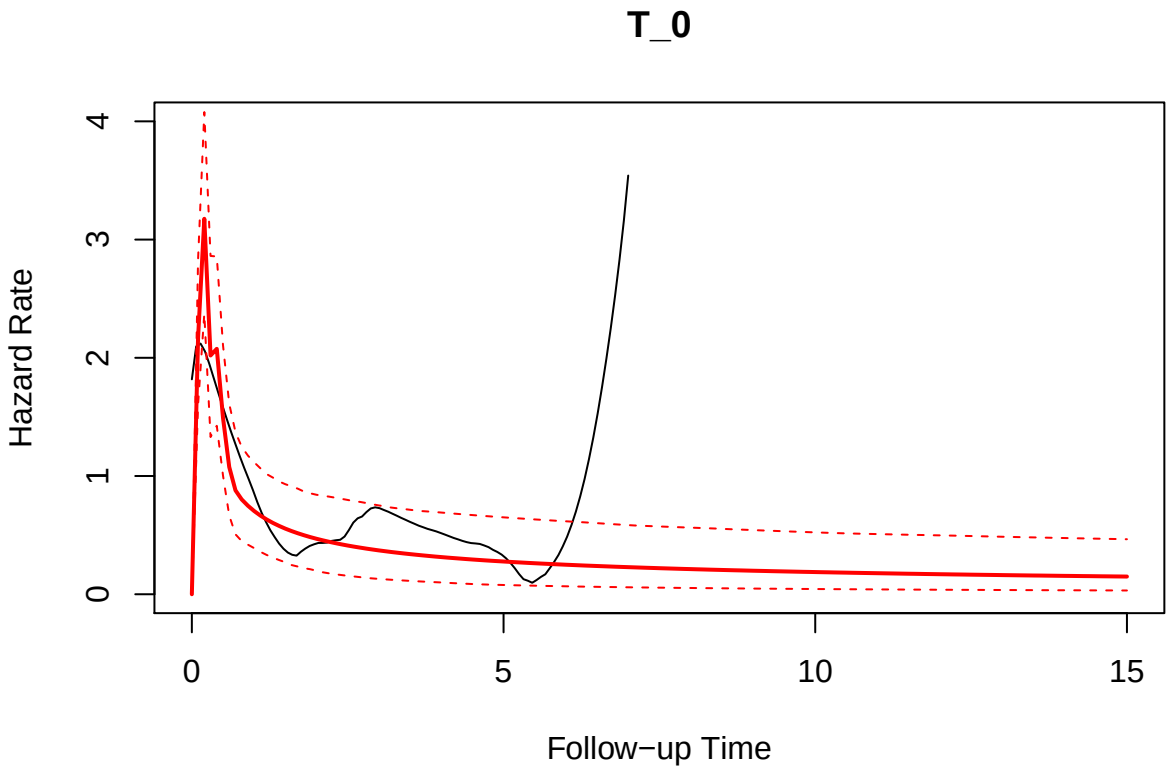

```
plot(fit_s0, main = "S_0", type = "hazard", t = seq(0, 15, 0.1),
     xlim = c(0, 15), ylim  =c(0, 4))
```




```
plot(fit_t0, main = "T_0", type = "survival", t = seq(0, 15, 0.1),
     xlim = c(0, 15))
abline(a = 0, b = 0, col = "gray")
```




```
plot(fit_t0, main = "T_0", type = "hazard", t = seq(0, 15, 0.1),
     xlim = c(0, 15), ylim  =c(0, 4))
```




```
predict(fit_s0, newdata = data[1,], type = "survival", times = 15)
```

```
## # A tibble: 1 x 2
##   .time   .pred
##   <dbl>   <dbl>
## 1    15 0.00917
```

```
predict(fit_t0, newdata = data[1,], type = "survival", times = 15)
```

```
## # A tibble: 1 x 2
##   .time   .pred
##   <dbl>   <dbl>
## 1    15 0.00616
```

```
predict(fit_s0, newdata = data[1,], type = "survival", times = 25)
```

```
## # A tibble: 1 x 2
##   .time   .pred
##   <dbl>   <dbl>
## 1    25 0.00351
```

```
predict(fit_t0, newdata = data[1,], type = "survival", times = 25)
```

```
## # A tibble: 1 x 2
##   .time   .pred
##   <dbl>   <dbl>
## 1    25 0.00170
```

```
predict(fit_s0, newdata = data[1,], type = "survival", times = 30)
```

```
## # A tibble: 1 x 2
##   .time   .pred
##   <dbl>   <dbl>
## 1    30 0.00238
```

```
predict(fit_t0, newdata = data[1,], type = "survival", times = 30)
```

```
## # A tibble: 1 x 2
##   .time    .pred
##   <dbl>    <dbl>
## 1    30 0.000999
```

```
cop_fit_ctrl = fit_copula(data = new_data[new_data$Treat == 0,], inits = 2.5)
```

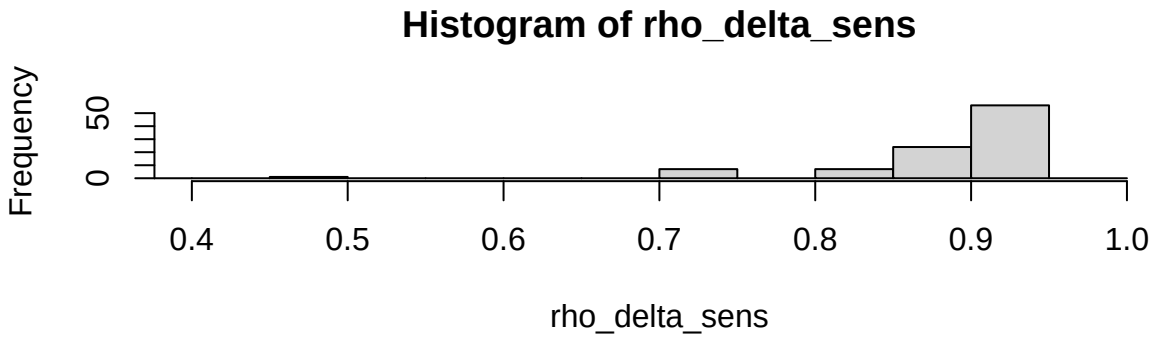
```
## iter    2 value -215.251196
## iter    4 value -215.258230
## final  value -215.258230
## converged
```

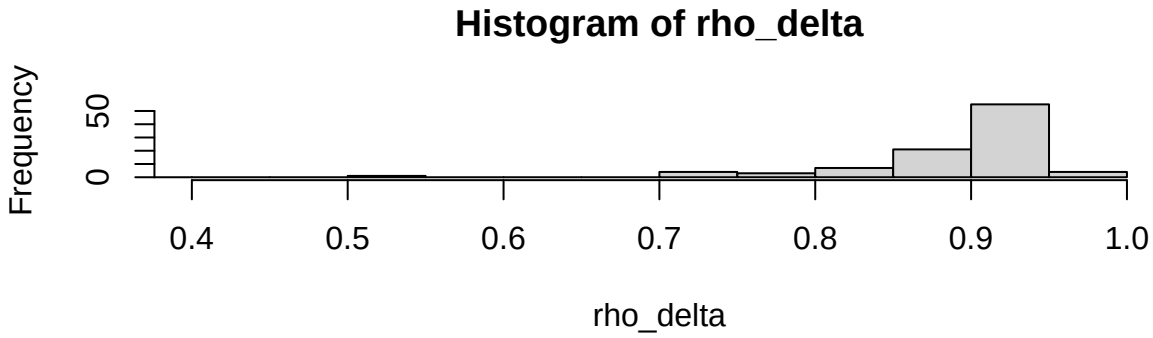
```
cop_fit_trt = fit_copula(data = new_data[new_data$Treat == 1,], inits = 2.5)
```

```
## iter    2 value -232.837522
## iter    4 value -236.121894
## iter    6 value -236.122286
## final  value -236.122286
## converged
```

```
rho13 = (exp(cop_fit_ctrl$par[1]) - 1)/(exp(cop_fit_ctrl$par[1]) + 1)
rho24 = (exp(cop_fit_trt$par[1]) - 1)/(exp(cop_fit_trt$par[1]) + 1)
```

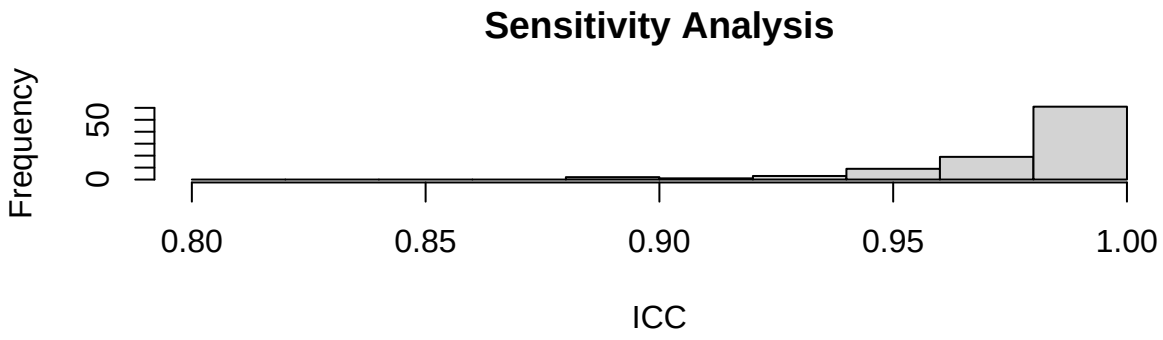
```
grid = seq(-1, 1, 0.2)
Pos.Def.Matrices(T0S0 = rho13, T1S1 = rho24,
                 T0T1 = grid, S0S1 = grid, T1S0 = grid, T0S1 = grid)
# Generated.Matrices = readRDS(file = "many_matrices.RData")
posdef_gen_matrices = Generated.Matrices[Generated.Matrices$Pos.Def.Status == 1,]
# saveRDS(object = Generated.Matrices, file = "many_matrices.RData")
rho_delta_sens = numeric()
for(i in 1:nrow(posdef_gen_matrices)){
  num = posdef_gen_matrices[i, "T0S0"] + posdef_gen_matrices[i, "T1S1"] - posdef_gen_matrices[i, "T1S0"]
  denom = 2*sqrt((1 - posdef_gen_matrices[i, "T0T1"])*
                   (1 - posdef_gen_matrices[i, "S0S1"]))
  rho_delta_sens = c(rho_delta_sens, num/denom)
}
par(mfrow = c(2, 1))
hist(rho_delta_sens, xlim = c(0.4, 1), breaks = seq(0.30, 1, 0.05))
hist(rho_delta, xlim = c(0.4, 1), breaks = seq(0.30, 1, 0.05))
```

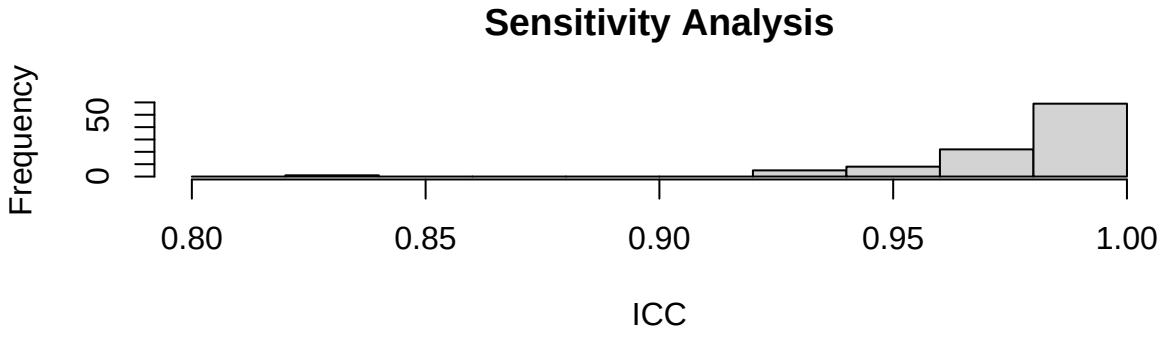




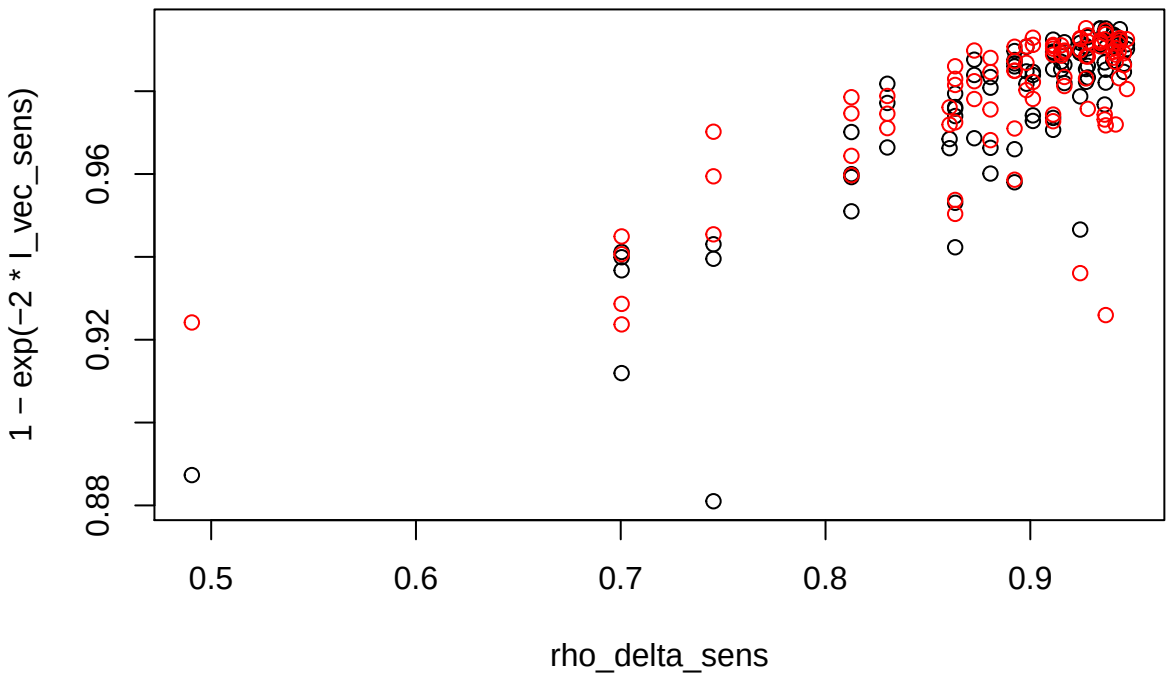

```
posdef_Sigma_list = as.list(1:nrow(posdef_gen_matrices))
for(i in 1:nrow(posdef_gen_matrices)){
  rho12 = posdef_gen_matrices[i, "S0S1"]
  rho13 = posdef_gen_matrices[i, "T0S0"]
  rho14 = posdef_gen_matrices[i, "T1S0"]
  rho23 = posdef_gen_matrices[i, "T0S1"]
  rho24 = posdef_gen_matrices[i, "T1S1"]
  rho34 = posdef_gen_matrices[i, "T0T1"]
  Sigma_temp = matrix(c(1, rho12, rho13, rho14,
                        rho12, 1, rho23, rho24,
                        rho13, rho23, 1, rho34,
                        rho14, rho24, rho34, 1), nrow = 4)
  posdef_Sigma_list[[i]] = Sigma_temp
}
library(parallel)
source("information_theoretic_functions.R")
cl = makeCluster(5)
clusterExport(cl, ls())
clusterEvalQ(cl, library(mvtnorm))
clusterEvalQ(cl, library(purrr))
clusterEvalQ(cl, library(flexsurv))
clusterEvalQ(cl, source("information_theoretic_functions.R"))
I_vec_sens = unlist(parLapply(cl = cl, fun = r_h, X = posdef_Sigma_list,
                              n = 1000))
saveRDS(I_vec_sens, file = "I_vec_sens.RData")
stopCluster(cl)
```

```
I_vec_sens = readRDS(file = "I_vec_sens.RData")
par(mfrow = c(2, 1))
hist(1 - exp(-2*I_vec_sens), xlab = "ICC", main = "Sensitivity Analysis",
     xlim = c(0.80, 1), breaks = seq(0.80, 1, 0.02))
hist(1 - exp(-2*I_vec), xlab = "ICC", main = "Sensitivity Analysis",
     xlim = c(0.80, 1), breaks = seq(0.80, 1, 0.02))
```

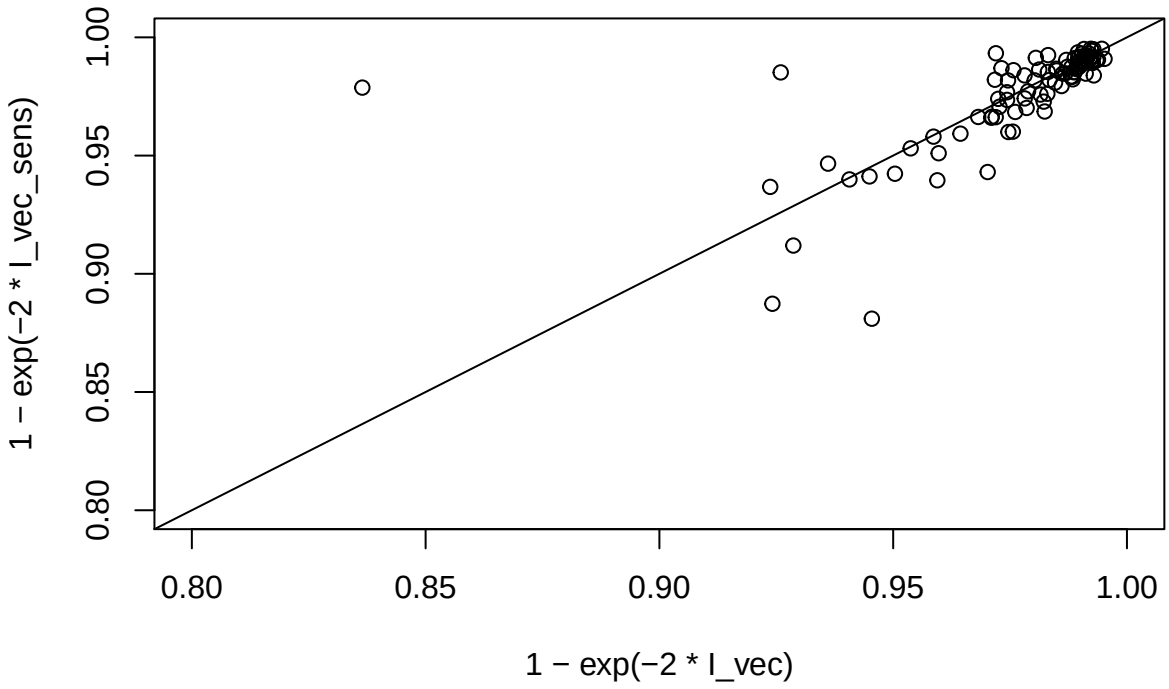





```
par(mfrow = c(1,1))
plot(rho_delta_sens, 1 - exp(-2*I_vec_sens))
points(rho_delta_sens, 1 - exp(-2*I_vec), col = "red")
```




```
plot(1 - exp(-2*I_vec), 1 - exp(-2*I_vec_sens),
     xlim = c(0.8, 1), ylim = c(0.8, 1))
abline(a = 0, b = 1)
```


1 − exp(−2 * l_vec_sens)
0.80
0.85
0.90
0.95
1.00
1 − exp(−2 * l_vec)
0.80
0.85
0.90
0.95
1.00