# Solutions: Chapter 10

## Exercise 1

a) `block.cp.init`: Competing risks experiment. Used only for the first iteration of the algorithm

```
> block.cp.init <- function(n, a1, a2) {
+
+     time <- rexp(n, a1 + a2)
+     to <- rbinom(n, 1, a1 / (a1 + a2))
+     to <- ifelse(to == 0, 3, to)
+     from <- rep(0, n)
+     id <- 1:n
+     data.frame(id = id, entry = rep(0, n), exit = time,
+                from = from, to = to)
+ }
```

`block.cp`: competing risks experiment with constant hazards

```
> block.cp <- function(data, a1, a2) {
+
+     n <- nrow(data)
+     time <- rexp(n, a1 + a2) + data$exit
+     entry <- data$exit
+     to <- rbinom(n, 1, a1 / (a1 + a2))
+     data.frame(id = data$id, entry = data$exit, exit = time,
+                from = data$to, to = to)
+ }
```

Function to simulate one data set following model of Exercise 1, Chapter 10 of the book:
**Input:**

`ll`: for the recursion. Leave it as it is.

`n`: sample size.

`h**`: transition hazards.

`cens.param`: a vector of 2 elements giving parameters for uniformly distributed censoring times.

```
> simul.chap10 <- function(ll = list(), n, h01, h03, h12, h13, h21, h23,
+                          cens.param) {
+
+     ## start in state 0
+     if (length(ll) == 0) {
+         ll[[1]] <- block.cp.init(n, h01, h03)
+         simul.chap10(ll, n, h01, h03, h12, h13, h21, h23, cens.param)
+     }
+
+     ## then further in the model
+     tmp <- ll[[length(ll)]]
+     if (!all(tmp$to == 3)) {
+         tab <- table(tmp$to)
+         transient <- names(tab)[names(tab) != "3"]
+         ll[[length(ll) + 1]] <- switch(transient,
+                    "1" = {
+                        aa <- block.cp(subset(tmp, to == 1), h12, h13)
+                        aa$to <- ifelse(aa$to == 1, 2, 3)
+                        aa
+                    },
+                    "2" = {
+                        bb <- block.cp(subset(tmp, to == 2), h21, h23)
+                        bb$to <- ifelse(bb$to == 1, 1, 3)
+                        bb
+                    })
+         simul.chap10(ll, n, h01, h03, h12, h13, h21, h23, cens.param)
+
+     } else {
+         almostThere <- do.call(rbind, ll)
+
+         ## time to deal with independent censoring
+         cens.times <- runif(n, cens.param[1], cens.param[2])
+         almostThere <- almostThere[order(almostThere$id, almostThere$exit), ]
+         cens.times <- cens.times[almostThere$id]
+         indic1 <- almostThere$exit < cens.times
+         indic2 <- almostThere$entry < cens.times
+         almostThere[indic2 != indic1, "exit"] <-
+             cens.times[indic2 != indic1]
+         almostThere[indic2 != indic1, "to"] <- "cens"
+         finito <- almostThere[indic2, ]
+
+         return(finito)
+     }
+ }
```

**Scenario 1**

```
> set.seed(4239)
> dat1.z0 <- simul.chap10(list(), 300, 1.2, 0.5, 0.7, 0.8, 1.2, 0.5,
+                         cens.param = c(0, 5))
> dat1.z1 <- simul.chap10(list(), 300, exp(0.1) * 1.2, exp(0.3) * 0.5,
+                         0.7, 0.8, exp(0.1) * 1.2, exp(0.3) * 0.5,
+                         cens.param = c(0, 5))
> dat1 <- rbind(dat1.z0, dat1.z1)
> dat1$Z <- c(rep(0, nrow(dat1.z0)), rep(1, nrow(dat1.z1)))
> dat1$id <- c((1:300)[dat1.z0$id], (301:600)[dat1.z1$id])
```

**Scenario 2**

```
> set.seed(4239)
> dat2.z0 <- simul.chap10(list(), 300, 1.2, 0.5, 0.7, 0.8, 2, 0.6,
+                         cens.param = c(0, 5))
> dat2.z1 <- simul.chap10(list(), 300, exp(0.1) * 1.2, exp(0.3) * 0.5,
+                         0.7, 0.8, exp(0.1) * 2, exp(0.3) * 0.6,
+                         cens.param = c(0, 5))
> dat2 <- rbind(dat2.z0, dat2.z1)
> dat2$Z <- c(rep(0, nrow(dat2.z0)), rep(1, nrow(dat2.z1)))
> dat2$id <- c((1:300)[dat2.z0$id], (301:600)[dat2.z1$id])
```

b) Analysis assuming an illness-death model

- From state 0 to state 1 for 1st scenario

```
> cox.scenar1.01 <- coxph(Surv(entry, exit, to == 1) ~ Z + cluster(id),
+                         dat1, subset = from %in% c(0, 2))
```

- From state 0 to state 3 for 1st scenario

```
> cox.scenar1.03 <- coxph(Surv(entry, exit, to == 3) ~ Z + cluster(id),
+                         dat1, subset = from %in% c(0, 2))
```

- From state 0 to state 1 for 2nd scenario

```
> cox.scenar2.01 <- coxph(Surv(entry, exit, to == 1) ~ Z + cluster(id),
+                         dat2, subset = from %in% c(0, 2))
```

- From state 0 to state 3 for 2nd scenario

```
> cox.scenar2.03 <- coxph(Surv(entry, exit, to == 3) ~ Z + cluster(id),
+                         dat2, subset = from %in% c(0, 2))
```

c) Distinguishing states 0 and 3

**Scenario 1**

```
> cox.c.scenar1.01 <- coxph(Surv(entry, exit, to == 1) ~ Z + cluster(id),
+                           dat1, subset = from == 0)
> cox.c.scenar1.21 <- coxph(Surv(entry, exit, to == 1) ~ Z + cluster(id),
+                           dat1, subset = from == 2)
> cox.c.scenar1.03 <- coxph(Surv(entry, exit, to == 3) ~ Z + cluster(id),
+                           dat1, subset = from == 0)
> cox.c.scenar1.23 <- coxph(Surv(entry, exit, to == 3) ~ Z + cluster(id),
+                           dat1, subset = from == 2)
```

**Scenario 2**

```
> cox.c.scenar2.01 <- coxph(Surv(entry, exit, to == 1) ~ Z + cluster(id),
+                           dat1, subset = from == 0)
> cox.c.scenar2.21 <- coxph(Surv(entry, exit, to == 1) ~ Z + cluster(id),
+                           dat1, subset = from == 2)
> cox.c.scenar2.03 <- coxph(Surv(entry, exit, to == 3) ~ Z + cluster(id),
+                           dat1, subset = from == 0)
> cox.c.scenar2.23 <- coxph(Surv(entry, exit, to == 3) ~ Z + cluster(id),
+                           dat1, subset = from == 2)
```

d) Simulation study
Main simulation function:
**Input**:

`N`: number of replications

**NOTE**: meant to be used in a `lapply()` call
**Output**:
A list of data frames. Each data frame contains the coeffient and the 2 variance estimates for the 6 Cox of models of interest

```
> main.simul <- function(N) {
+
+     print(N)
+     ## simulate scenar1
+     ## simulate baseline data
+     base.dat <- simul.chap10(list(), n0, h01, h03, h12, h13, h01, h03,
+                              cens.param)
+     base.dat$cov <- 0
+     ## simulate data for Z = 1
+     cov.dat <- simul.chap10(list(), n1, exp(beta01) * h01, exp(beta03) * h03,
+                             h12, h13, exp(beta01) * h01, exp(beta03) * h03,
+                             cens.param)
+     cov.dat$cov <- 1
+     dat1 <- rbind(base.dat, cov.dat)
+     dat1$id <- c((1:n0)[base.dat$id], ((n0 + 1):(n1 + n0))[cov.dat$id])
```

```
+
+     ## simulate scenar2
+     ## simulate baseline data
+     base.dat <- simul.chap10(list(), n0, h01, h03, h12, h13, h21, h23,
+                              cens.param)
+     base.dat$cov <- 0
+     ## simulate data for Z = 1
+     cov.dat <- simul.chap10(list(), n1, exp(beta01) * h01, exp(beta03) * h03,
+                             h12, h13, exp(beta01) * h21, exp(beta03) * h23,
+                             cens.param)
+     cov.dat$cov <- 1
+     dat2 <- rbind(base.dat, cov.dat)
+     dat2$id <- c((1:n0)[base.dat$id], ((n0 + 1):(n1 + n0))[cov.dat$id])
+
+ ### fit of the cox models
+     res <- lapply(1:2, function(i) {
+
+         dd <- as.name(paste("dat", i, sep = ""))
+
+         ## grouping states 0 and 2
+         cox.naive.01 <- coxph(Surv(entry, exit, to == 1) ~ cov + cluster(id),
+                               eval(dd), subset = from %in% c(0, 2))
+         cox.naive.03 <- coxph(Surv(entry, exit, to == 3) ~ cov + cluster(id),
+                               eval(dd), subset = from %in% c(0, 2))
+
+         ## not grouping
+         cox.ng.01 <- coxph(Surv(entry, exit, to == 1) ~ cov + cluster(id),
+                            eval(dd), subset = from == 0)
+         cox.ng.21 <- coxph(Surv(entry, exit, to == 1) ~ cov + cluster(id),
+                            eval(dd), subset = from == 2)
+         cox.ng.03 <- coxph(Surv(entry, exit, to == 3) ~ cov + cluster(id),
+                            eval(dd), subset = from == 0)
+         cox.ng.23 <- coxph(Surv(entry, exit, to == 3) ~ cov + cluster(id),
+                            eval(dd), subset = from == 2)
+
+         coefs <- c(coef(cox.naive.01), coef(cox.naive.03),
+                    coef(cox.ng.01), coef(cox.ng.21),
+                    coef(cox.ng.03), coef(cox.ng.23))
+         var.simple <- c(cox.naive.01$naive.var[1, 1],
+                         cox.naive.03$naive.var[1, 1],
+                         cox.ng.01$naive.var[1, 1],
+                         cox.ng.21$naive.var[1, 1],
+                         cox.ng.03$naive.var[1, 1],
+                         cox.ng.23$naive.var[1, 1])
+         var.robust <- c(diag(vcov(cox.naive.01)), diag(vcov(cox.naive.03)),
+                         diag(vcov(cox.ng.01)), diag(vcov(cox.ng.21)),
```

```
+                         diag(vcov(cox.ng.03)), diag(vcov(cox.ng.23)))
+
+         data.frame(coef = coefs, var.simple = var.simple,
+                    var.robust = var.robust)
+     })
+
+     res
+ }
```

Run the simulations

```
> n0 <- n1<- 200
> h01 <- 1.2
> h03 <- 0.5
> h12 <- 0.7
> h13 <- 2
> h21 <- 2
> h23 <- 1
> beta01 <- 0.5
> beta03 <- 0.3
> cens.param <- c(0, 5)
> set.seed(88749)
> res.global.chap10 <- lapply(1:1000, main.simul)
```

Extract the parts we need:

```
> scenar1 <- lapply(res.global.chap10, "[[", 1) ## illness-death model
> scenar2 <- lapply(res.global.chap10, "[[", 2) ## misspecified one
```

A little function that gives a list per Cox model

```
> organise <- function(res) {
+     lapply(1:6, function(i) {
+         tmp <- lapply(res, function(ll) {
+             ll[i, ]
+         })
+         do.call(rbind, tmp)
+     })
+ }
> scenar1.org <- organise(scenar1)
> scenar2.org <- organise(scenar2)
```

Boxplots of the regression coefficients

```
> titles <- c(expression("healthy" %->% "diseased"),
+             expression("healthy" %->% "death"),
+             expression("healthy" %->% "diseased"),
+             expression("healthy after 1st recov." %->% "diseased"),
+             expression("healthy" %->% "death"),
```

```
+             expression("healthy after 1st recov." %->% "death"))
> par(mfrow = c(3, 2))
> for (i in 1:6) {
+     boxplot(cbind(scenar1.org[[i]]$coef,
+                   scenar2.org[[i]]$coef),
+             main = titles[i], cex.main = 1.6,
+             names = c("1st scenario", "2nd scenario"))
+ }
```

Comparison of the variance estimates with empirical variance

```
> empVar1 <- sapply(scenar1.org, function(ll) {var(ll$coef)})
> empVar2 <- sapply(scenar2.org, function(ll) {var(ll$coef)})
> av.scenar1 <- t(sapply(scenar1.org, function(ll) {
+     apply(ll, 2, mean)
+ }))
> av.scenar1 <- cbind(av.scenar1, empVar1)
> av.scenar2 <- t(sapply(scenar2.org, function(ll) {
+     apply(ll, 2, mean)
+ }))
> av.scenar2 <- cbind(av.scenar2, empVar2)
```