Solutions: Chapter 9

Exercise 1

We use the functions of chapter 8 adding censoring:

```
> set.seed(989173)
> dat.chap9.exo1 <- sim.chap8.exo2(list(), 200, 0.4,
+                                  ch.from.0, ch.from.1,
+                                  prob.from.0, prob.from.1,
+                                  cens = list(yes = TRUE,
+                                  param = c(0, 3)))
```

Computation of the cumulative transition hazards using **mvna**

```
> ## the possible transitions
> tra.idm <- matrix(FALSE, nrow = 3, ncol = 3)
> tra.idm[1, 2:3] <- TRUE
> tra.idm[2, c(1, 3)] <- TRUE
> ## estimation
> mvna.chap9.exo1 <- mvna(dat.chap9.exo1, c("0", "1", "2"), tra.idm, "cens")
```

Plot of the Nelson-Aalen estimates

```
> xyplot(mvna.chap9.exo1, layout = c(2, 2), lwd = 2,
+        strip = strip.custom(bg = "white",
+        factor.levels = c(expression(0 %->% 1),
+        expression(0 %->% 2),
+        expression(1 %->% 0),
+        expression(1 %->% 2)),
+        par.strip.text = list(cex = 1.1,
+        font = 2)))
```

True quantities

```
> xx <- seq(0, 4, 0.001)
```

Transition $0 \to 1$

```
> trellis.focus("panel", 1, 1, highlight = FALSE)
> llines(xx, (1/9) * xx^3 + (1/5) * xx, col = "red", lwd = 2)
```

Transition $0 \to 2$

```
> trellis.focus("panel", 2, 1, highlight = FALSE)
> llines(xx, 3 * log(xx + 1), col = "red", lwd = 2)
```

Transition $1 \to 0$

```
> trellis.focus("panel", 1, 2, highlight = FALSE)
> llines(xx, xx, col = "red", lwd = 2)
```

Transition $1 \to 2$

```
> trellis.focus("panel", 2, 2, highlight = FALSE)
> llines(xx, (4/3) * (xx)^(3/2), col = "red", lwd = 2)
```

Exercise 2

Estimation of the cumulative transition hazards in a 'univariate' fashion.
a) First, we have to modify the data. We will only keep transitions out of state 1.

```
> dat.chap9.exo2 <- subset(dat.chap9.exo1, from == 1)
```

Then, we censor transitions into state 0,

```
> dat.chap9.exo2$to <- with(dat.chap9.exo2,
+                           ifelse(to == 0, "cens", to))
```

and then use `mvna`

```
> tra.univ <- matrix(FALSE, ncol = 2, nrow = 2)
> tra.univ[1, 2] <- TRUE
> ##
> mvna.univ <- mvna(dat.chap9.exo2, c("1", "2"), tra.univ, "cens")
```

Let's check our computation:

```
> plot(mvna.univ)
> lines(mvna.chap9.exo1, tr.choice = "1 2", col = 2)
```

b) Using `survfit`, we can keep the original data and use the subset argument.

```
> surv.12 <- survfit(Surv(entry, exit, to == 2) ~ 1,
+                    dat.chap9.exo1, subset = from == 1)
```

Nelson-Aalen estimator:

```
> na.surv.12 <- cumsum(surv.12$n.event[-1] / surv.12$n.risk[-1])
```

and check (keep the last plot open)

```
> lines(surv.12$time[-1], na.surv.12, col = 4, type = "s", lwd = 2)
```

Exercise 3

Landmark plots:
We first define the time points used for landmarking

```
> landmark.times <- seq(0.2, 1.2, 0.2)
```

Compute the transition probabilities for the different landmarks:

```
> landmark.etm <- lapply(landmark.times, function(start.time) {
+     etm(dat.chap9.exo1, c("0", "1", "2"), tra.idm, "cens",
+         start.time)
+ })
```

and display the result

```
> nf <- layout(matrix(1:6, nrow = 2, byrow = TRUE), width = rep(1, 10),
+              height = rep(1, 10))
> for (i in seq_along(landmark.times)) {
+     plot(landmark.etm[[i]], tr.choice = "0 2", lwd = 2, legend = FALSE,
+          main = paste(expression("s ="), landmark.times[i], sep = " "),
+          xlim = c(0, 100), lty = 2)
+     lines(landmark.etm[[i]], tr.choice = "1 2", lwd = 2, col = 1,
+           lty = 1, xlim = c(0, 100))
+     if (i == 1) {
+         legend("bottomright",
+                c(expression(P["02"](s, t)), expression(P["12"](s, t))),
+                col = 1, lty = c(2, 1), lwd = 2, bty = "n",
+                cex = 1.3)
+     }
+ }
```

Exercise 4

We need to rewrite the function of chapter 8 to include a 'third' competing risk. The code for censored observation is `13` in the column `to` of the resulting data frame.

```
> block.cp.cens <- function(data, sum.ch, probs = list(),
+                           from, max.int = 500) {
+
+     n <- nrow(data)
+     inc <- 1; n.failure <- 0
+     stime <- numeric(n)
+
+     while (inc <= n) {
+         u <- runif(1)
+         if (sum.ch(0, log(1 - u)) * sum.ch(max.int, log(1 - u)) < 0) {
```

```
+             res <- uniroot(sum.ch, c(0, max.int), tol = 0.0001, y = log(1 - u))
+             stime[inc] <- res$root
+             inc <- inc + 1
+         }
+         else {
+             n.failure <- n.failure + 1
+         }
+     }
+     time <- stime + data$exit
+
+     to <- numeric(n)
+     for (i in seq_len(n)) {
+         to[i] <- which(t(rmultinom(1, 1, c(probs[[1]](time[i]),
+                                            probs[[2]](time[i]),
+                                            probs[[3]](time[i])))) == 1)
+     }
+     to <- switch(from,
+                  "0" = {
+                      ifelse(to == 3, 13, to)
+                  },
+                  "1" = {
+                      ifelse(to == 3, 13, (to == 1) * 0 + (to == 2) * 2)
+                  })
+
+     data.frame(id = data$id, entry = data$exit, exit = time,
+                from = data$to, to = to)
+ }
```

`ch.from.*.cens`: cumulative transition hazards (including censoring)

```
> ch.from.0.cens <- function(x, y) {
+     return((1/9) * x^3 + (1 / 5) * x + 3 * log(x + 1) + 0.9 * x + y)
+ }
> ch.from.1.cens <- function(x, y) {
+         return(x + (4/3) * x^(3/2) + (3/4) * x^(4/3) + y)
+ }
```

`prob.from.*.cens` multinomial experiment that decides on the event type.

```
> prob.from.0.cens <- list(function(tt) {
+     return(((1/3) * tt^2 + (1/5)) /
+            (3 / (1 + tt) + (1/3) * tt^2 + 1/5 + 0.9))},
+                          function(tt) {
+     return((3 / (1 + tt)) /
+            (3 / (1 + tt) + (1/3) * tt^2 + 1/5 + 0.9))},
+                          function(tt) {
+     return(0.9 /
```

```
+            (3 / (1 + tt) + (1/3) * tt^2 + 1/5 + 0.9))})
> prob.from.1.cens <- list(function(tt) {
+     return(1 / (1 + 2 * sqrt(tt) + tt^(1/3)))},
+                          function(tt) {
+     return((2 * sqrt(tt)) / (1 + 2 * sqrt(tt) + tt^(1/3)))},
+                          function(tt) {
+     return((tt^(1/3)) / (1 + 2 * sqrt(tt) + tt^(1/3)))})
```

`sim.chap9.exo4`: Almost the same function as the one for chapter 8, except for the call to `block.cp.cens` instead of `block.cp`:

```
> sim.chap9.exo4 <- function(ll = list(), n, p.init,
+                            ch.from0, ch.from1,
+                            prob.from0, prob.from1,
+                            cens = list(yes = FALSE, param = NULL),
+                            max.int = 500) {
+
+     if (length(ll) == 0) {
+         ## create some empty data for a start
+         from <- to <- rbinom(n, size = 1, p.init)
+         id <- seq_len(n)
+         entry <- exit <- numeric(n)
+         temp <- data.frame(id, from, to, entry, exit)
+         ll[[1]] <- rbind(block.cp.cens(temp[temp$from == 0, ], ch.from0,
+                                        prob.from0, "0"),
+                          block.cp.cens(temp[temp$from == 1, ], ch.from1,
+                                        prob.from1, "1"))
+         sim.chap9.exo4(ll, n, p.init, ch.from0, ch.from1,
+                        prob.from0, prob.from1, cens,
+                        max.int)
+     }
+
+     ## then...
+     tmp <- ll[[length(ll)]]
+
+     if (!all(tmp$to == 2)) {
+         ll[[length(ll) + 1]] <- rbind(block.cp.cens(subset(tmp, to == 0),
+                                                     ch.from0,
+                                                prob.from0, "0"),
+                                       block.cp.cens(subset(tmp, to == 1),
+                                                     ch.from1,
+                                                prob.from1, "1"))
+         sim.chap9.exo4(ll, n, p.init, ch.from0, ch.from1,
+                        prob.from0, prob.from1, cens, max.int)
+     } else {
+         almostThere <- do.call(rbind, ll)
+
```

```
+         if (cens$yes) {
+             ## time to deal with independent censoring
+             cens.times <- runif(n, cens$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, ]
+         } else {
+             finito <- almostThere
+         }
+
+         return(finito)
+     }
+ }
> set.seed(400920)
> dat.chap9.exo4 <- sim.chap9.exo4(list(), 100, 0.1,
+                                  ch.from.0.cens, ch.from.1.cens,
+                                  prob.from.0.cens, prob.from.1.cens,
+                                  cens = list(yes = FALSE))
```

Estimation of the cumulative transition hazards

```
> mvna.chap9.exo4 <- mvna(dat.chap9.exo4, c("0", "1", "2"), tra.idm, "13")
```

Plot of the Nelson-Aalen estimates

```
> xyplot(mvna.chap9.exo4, layout = c(2, 2), lwd = 2,
+        strip = strip.custom(bg = "white",
+        factor.levels = c(expression(0 %->% 1),
+        expression(0 %->% 2),
+        expression(1 %->% 0),
+        expression(1 %->% 2)),
+        par.strip.text = list(cex = 1.1,
+        font = 2)))
```

True quantities

```
> xx <- seq(0, 4, 0.001)
```

Transition $0 \to 1$

```
> trellis.focus("panel", 1, 1, highlight = FALSE)
> llines(xx, (1/9) * xx^3 + (1/5) * xx, col = "red", lwd = 2)
```

Transition $0 \to 2$

```
> trellis.focus("panel", 2, 1, highlight = FALSE)
> llines(xx, 3 * log(xx + 1), col = "red", lwd = 2)
```

Transition $1 \to 0$

```
> trellis.focus("panel", 1, 2, highlight = FALSE)
> llines(xx, xx, col = "red", lwd = 2)
```

Transition $1 \to 2$

```
> trellis.focus("panel", 2, 2, highlight = FALSE)
> llines(xx, (4/3) * (xx)^(3/2), col = "red", lwd = 2)
```

Exercise 5

a)
`block.cp.const`: Competing risks experiment with constant hazards **Input**:

`data:` data in the **etm/mvna** format

`hazards:` list of 2: the cause specific hazards

`fail.cause:` How are coded the 2 resulting events

```
> block.cp.const <- function(data, hazards = list(), fail.cause) {
+
+     n <- nrow(data)
+
+     time <- data$exit + rexp(n, hazards[[1]] + hazards[[2]])
+     to <- rbinom(n, 1, hazards[[1]] / (hazards[[1]] + hazards[[2]]))
+     to <- ifelse(to == 0, fail.cause[2], fail.cause[1])
+     data.frame(id = data$id, entry = data$exit, exit = time,
+                from = data$to, to = to)
+ }
```

Function to simulate the data:
Input:

`n:` sample size

`hazards:` list of hazards, in the right order. The list must have names (see example)

```
> sim.chap9.exo5 <- function(n, hazards = list()) {
+
+     ## from state 0 for a start, then we loop (with a while)
+     exit <- rexp(n, hazards$"01" + hazards$"02")
+     to <- rbinom(n, size = 1,
+                  prob = hazards$"01" / (hazards$"01" + hazards$"02"))
```

```
+     to <- ifelse(to == 0, 2, 1)
+     id <- seq_len(n)
+     from <- entry <- rep(0, n)
+     ll <- list(data.frame(id, from, to, entry, exit))
+
+     transient <- seq(2, 6, 2)
+     i <- 1
+     while (!all(ll[[i]]$to %in% c(7, 8))) {
+         temp <- ll[[i]][ll[[i]]$to == transient[i], ]
+         if (nrow(temp) == 0) break
+         haz.tmp <- haz[grep(transient[i], names(hazards))]
+
+         ll[[i + 1]] <- block.cp.const(temp, haz.tmp,
+                                       fail.cause = c(transient[i] + 2,
+                                       transient[i] + 1))
+         i <- i + 1
+     }
+
+     do.call(rbind, ll)
+ }
```

Creation of the data set

```
> haz <- list(0.4, 0.5, 1.3, 0.9, 0.5, 1.4, 0.05, 0.07)
> names(haz) <- c("01", "02", "23", "24", "45", "46", "67", "68")
> ##
> set.seed(73222)
> dat.chap9.exo5 <- sim.chap9.exo5(500, haz)
```

b)

```
> ## matrix of logical defining the possible transitions
> tra.dli <- matrix(FALSE, 9, 9,
+               dimnames = list(as.character(0:8), as.character(0:8)))
> tra.dli[1, 2:3] <- TRUE
> tra.dli[3, 4:5] <- TRUE
> tra.dli[5, 6:7] <- TRUE
> tra.dli[7, 8:9] <- TRUE
```

Computation of the transition probabilities

```
> dli.etm <- etm(dat.chap9.exo5, as.character(0:8), tra.dli, NULL, s = 0)
```

Basic `xyplot` of all direct transitions

```
> xyplot(dli.etm, lwd = 2)
```

Estimation of the current leukemia-free survival and its variance

```
> clfs <- dli.etm$est["0", "0", ] + dli.etm$est["0", "6", ]
> var.clfs <- dli.etm$cov["0 0", "0 0", ] +
+     dli.etm$cov["0 6", "0 6", ] + 2 * dli.etm$cov["0 0", "0 6", ]
```

Plot with confidence interval

```
> ciplus <- clfs + qnorm(0.975) * sqrt(var.clfs)
> cimoins <- clfs - qnorm(0.975) * sqrt(var.clfs)
> plot(dli.etm$time, clfs, type = "s", bty = "n", ylim = c(0, 1),
+      xlab = "Year", ylab = "Probability")
> lines(dli.etm$time, cimoins, lty = 3, type = "s")
> lines(dli.etm$time, ciplus, lty = 3, type = "s")
```

c)
Load the data set from the journal's website

```
> con <- url("http://www.jstatsoft.org/v38/i04/supp/3")
> load(con)
> close(con)
```

Computation of the transition probabilities

```
> jss.etm <- etm(dli.data, as.character(0:8), tra.dli, "cens", s = 0)
```

Basic xyplot of all direct transitions

```
> xyplot(jss.etm, lwd = 2)
```

Estimation of the CLFS

```
> clfs <- jss.etm$est["0", "0", ] + jss.etm$est["0", "6", ]
> var.clfs <- jss.etm$cov["0 0", "0 0", ] +
+     jss.etm$cov["0 6", "0 6", ] + 2 * jss.etm$cov["0 0", "0 6", ]
```

Plot + confidence interval

```
> ciplus <- clfs + qnorm(0.975) * sqrt(var.clfs)
> cimoins <- clfs - qnorm(0.975) * sqrt(var.clfs)
> plot(jss.etm$time, clfs, type = "s", bty = "n", ylim = c(0, 1),
+      xlab = "Year", ylab = "Probability")
> lines(jss.etm$time, cimoins, lty = 3, type = "s")
> lines(jss.etm$time, ciplus, lty = 3, type = "s")
```