Solutions: Chapter 2

Exercise 1

Show that $\mathrm{P}(T \leq t) = 1 - \exp(-\int_0^t \alpha(u)du)$

Proof. $\alpha(t) = \frac{f(t)}{1-F(t)}$. Using the chain rule, we obtain

$$\int_0^t \alpha(u)du = -\log(1 - \mathrm{P}(T \leq t))$$
$$1 - \exp(-\int_0^t \alpha(u)du) = \mathrm{P}(T \leq t)$$

□

Exercise 2

Function that computes the cumulative hazard for Gompertz distribution. `time.points` is a vector of time points.

```
> A.gompertz <- function(time.points) {
+     2 * expm1(time.points / 2)
+ }
```

Next use the `prodint` function defined in the book:

```
> prodint <- function(time.points, A) {
+     times <- c(0, sort(unique(time.points)))
+     S <- prod(1 - diff(apply(X = matrix(times),
+                              MARGIN = 1, FUN = A)))
+     return(S)
+ }
```

Approximation using `prodint`:

```
> times <- seq(0, 1, 0.001)
> prodint(times, A.gompertz)
```

The true value is

```
> exp(-2 * expm1(max(times) / 2))
```

Exercise 3

Simulate Gompertz distributed survival times through the R-package `eha`

```
> require(eha)
```

Simulation of 100 individuals

```
> set.seed(7720)
> event.time <- rgompertz(100, 1, 2)
```

Simulation of censoring times and creation of a dataframe

```
> cens.time <- runif(100, 0, 2.5)
> obs.time <- pmin(event.time, cens.time)
> status <- as.numeric(obs.time == event.time)
> dat.exo3 <- data.frame(obs.time, status)
```

Survival probability using the `survfit()` function

```
> require(survival)
> S <- survfit(Surv(obs.time, status) ~ 1, dat.exo3)
> S$surv[S$time > 1][1]
```

Estimation with the help of `prodint`

```
> with(dat.exo3, prodint(obs.time[obs.time <= 1], A.gompertz))
```

Compution of the Nelson-Aalen estimator

```
> nelaal <- cumsum(S$n.event / S$n.risk)
```

Comparison with the true cumulative hazard

```
> plot(S$time, nelaal, type = "s")
> xx <- seq(0, 2, 0.001)
> lines(xx, 2 * expm1(xx/2), type = "l", col = 2)
> legend("topleft", c("Nelson-Aalen estimates",
+                     "True cumulative hazard"),
+                     col = c(1, 2), lty = 1, bty = "n")
```

Exercise 4

Creation of the counting process using the `survfit` object `S` created above:

```
> nt <- cumsum(S$n.event)
```

Computation of the compensator, using the `approxfun` function

```
> YtimesHaz <- c(0, S$n.risk * exp(S$time/2))
> ff <- approxfun(c(0, S$time), YtimesHaz)
> xx <- seq(0, round(max(S$time), 2), 0.01)
> compensator <- numeric(length(xx))
> for (i in seq_along(compensator)) {
+     compensator[i] <- integrate(ff, 0, xx[i], subdivisions = 1000)$value
+ }
```

Plot the compensator and counting process

```
> plot(x = c(0, 2), y = c(0, 80),
+      xlab = expression(paste(Time, " ", italic(t))), ylab = "",
+      type="n", axes = F, main = "", cex.main = 1.7, cex.lab = 1.5)
> axis(1, at = seq(0, 2, 0.5), cex.axis = 1.5)
> axis(2, at = seq(0, 80, 10), cex.axis = 1.25)
> box()
> lines(c(0, S$time), c(0, nt), type="s", lwd=2)
> lines(xx, compensator, type = "l", lwd = 2)
```

Exercise 5

With at least 50% censoring

```
> set.seed(9420)
> event.time <- rgompertz(100, 1, 2)
> cens.time <- runif(100, 0, 1.2)
> obs.time <- pmin(event.time, cens.time)
> status <- as.numeric(obs.time == event.time)
> dat.exo5 <- data.frame(obs.time, status)
```

Survival probability using `survfit()`

```
> S <- survfit(Surv(obs.time, status) ~ 1, dat.exo5)
> S$surv[S$time > 1][1]
```

And using `prodint()`

```
> with(dat.exo5, prodint(obs.time[obs.time <= 1], A.gompertz))
```

Exercise 6

Simulation of left-truncation times:

```
> set.seed(662344)
> dat.exo3$trunc.time <- rweibull(100, 1/4, 1/3)
> ## enter in the sample only those individuals whose truncation times
> ## are smaller than their survival times
> dat.exo3.tr <- dat.exo3[dat.exo3$trunc.time < dat.exo3$obs.time, ]
```

Computation of the Nelson-Aalen estimator using survival:

```
> s.exo6 <- survfit(Surv(trunc.time, obs.time, status) ~ 1,
+                   dat.exo3.tr)
> na.exo6 <- cumsum(s.exo6$n.event[s.exo6$n.risk > 0] /
+                   s.exo6$n.risk[s.exo6$n.risk > 0])
```

Plot of the true quantity:

```
> xx <- seq(0, 2, 0.001)
> plot(xx, 2 * expm1(xx / 2), type = "l")
> lines(s.exo6$time[s.exo6$n.risk > 0], na.exo6, type = "s")
```

Exercise 7

Simulation of data:
Input:

`n`: required sample size

`h01, h02`: cause-spcific hazards

`cens.param`: Parameters for simulating the censoring times

```
> simul.dat.cp <- function(n, h01, h02, cens.param) {
+
+     times <- rexp(n, h01 + h02)
+     ev <- rbinom(n, size = 1, prob = h01 / (h01 + h02))
+     ev <- ifelse(ev == 0, 2, 1)
+
+     cens.time <- runif(n, cens.param[1], cens.param[2])
+
+     obs.time <- pmin(times, cens.time)
+     obs.cause <- as.numeric(times == obs.time) * ev
+
+     data.frame(obs.time, obs.cause)
+ }
> set.seed(923)
> dat.exo7 <- simul.dat.cp(200, 0.5, 0.9, c(0.5, 3))
```

We compute the Nelson-Aalen estimates using `survfit()`
That's just to get the number of events and the risk set

```
> temp01 <- survfit(Surv(obs.time, obs.cause == 1) ~ 1, dat.exo7)
> temp02 <- survfit(Surv(obs.time, obs.cause == 2) ~ 1, dat.exo7)
> na01 <- cumsum(temp01$n.event / temp01$n.risk)
> na02 <- cumsum(temp02$n.event / temp02$n.risk)
```

Exercise 8

Show that

$$\mathrm{P}(T \leq t, X_T = 1) \leq 1 - \exp\left(-\int_0^t \alpha_{01}(u)du\right)$$

Proof. We note that

$$\begin{aligned} P(T > t) &= \exp(-\int_0^t \alpha_{01}(u)du - \int_0^t \alpha_{02}(u)du) \\ &= \exp(-\int_0^t \alpha_{01}(u)du) \cdot \exp(-\int_0^t \alpha_{02}(u)du) \\ &\leq \exp(-\int_0^t \alpha_{01}(u)du). \end{aligned}$$

$$\begin{aligned} \mathrm{P}(T \leq t, X_T = 1) &= \int_0^t \mathrm{P}(T > u-)\alpha_{01}(u)du \\ &\leq \int_0^t \exp(-\int_0^t \alpha_{01}(u)du)\alpha_{01}(u)du = 1 - \exp\left(-\int_0^t \alpha_{01}(u)du\right). \end{aligned}$$

□

Check overestimation

```
> ## cause-specific survivor function
> csSurv <- temp01$surv
> ## cumulative incidence function using survfit
> ## in the survival package (from version 2.35)
> cif <- survfit(Surv(obs.time, obs.cause != 0) ~ 1, dat.exo7,
+                etype = obs.cause)
> ## plots
> plot(temp01$time, 1 - csSurv, type = "s", xlab = "time", ylab = "")
> lines(cif[1], fun = "event", mark.time = FALSE, col = 2)
> legend("topleft", c("Cause-specific survivor function",
+                     "Cumulative incidence function"),
+        col = c(1, 2), lty = 1, bty = "n")
```

Exercise 9

Function to simulate one data set:
Input:

`n:` sample size

`h01, h02, h12:` hazards

`cens:` one of `"none"`, `"independent"` and `"state"`. `"state"` considers state dependent censoring

`cens.param:` list of 2 elements:

- `first:` for independent censoring, parameters of the uniform censoring distribution, keep `second` to `NULL`. For state dependent censoring, parameters for censoring in state 0.
- `second:` parameter for censoring in state 1

`trun:` a list of three elements:

- `yes:` Logical. Whether to have truncation
- `shape, scale:` shape and scale parameters for the gamma truncation time generation

Output:
A data frame in the **mvna/etm** format, i.e., with columns `(id, entry, exit, from, to)`. Censored observation are marked as `"13"` in `to`.

```
> simul.chap2.exo9 <- function(n, h01, h02, h12, ## sample size and hazards
+                              cens = c("none", "random", "state"),
+                              cens.param = list(first = NULL, second = NULL),
+                              trun = list(yes = FALSE, shape = NULL, scale = NULL))
+ {
+     cens <- match.arg(cens)
+
+     ## simulate data for exit from state 0 (+- same as exo 7)
+     id <- seq_len(n)
+     exit <- rexp(n, h01 + h02)
+     to <- rbinom(n, size = 1, prob = h01 / (h01 + h02))
+     to <- ifelse(to == 0, 2, 1)
+     from <- entry <- rep(0, n)
+
+     if (cens == "state") {
+         cens.time <- runif(n, cens.param$first[1], cens.param$first[2])
+         exit <- pmin(exit, cens.time)
+         to <- as.numeric(exit != cens.time) * to
+         to <- ifelse(to == 0, 13, to) ## put the censored obs at 13
+     }
+
+     data1 <- data.frame(id, entry, exit, from, to)
+
+     ## from state 1
+     data2 <- data1[data1$to == 1, ]
+     n2 <- nrow(data2)
+     data2$entry <- data2$exit
```

```
+     data2$exit <- data2$exit + rexp(n2, h02)
+     data2$from <- 1
+
+     if (cens == "state") {
+         cens.time <- runif(n2, cens.param$second[1], cens.param$second[2])
+         data2$exit <- pmin(data2$exit, cens.time)
+         data2$to <- ifelse(data2$exit == cens.time, 13, 2)
+     } else {
+         data2$to <- 2
+     }
+
+     ## merge
+     data.idm <- rbind(data1, data2)
+
+     ## independent left-truncation
+     if (trun$yes) {
+         trun.time <- rgamma(n, scale = trun$scale, shape = trun$scale)
+         data.idm$trun.time <- trun.time[data.idm$id]
+         ind.exit <- data.idm$exit > data.idm$trun.time
+         ind.entry <- data.idm$entry > data.idm$trun.time
+         data.idm$entry[ind.exit & !ind.entry] <-
+             data.idm$trun.time[ind.exit & !ind.entry]
+         data.idm <- data.idm[ind.exit, ]
+     }
+
+     ## independent censoring
+     if (cens == "random") {
+         cens.times <- runif(n,
+                             cens.param$first[1], cens.param$first[2])
+         data.idm$cens.time <- cens.times[data.idm$id]
+         ind.cens.exit <- data.idm$cens.time < data.idm$exit
+         ind.cens.entry <- data.idm$cens.time < data.idm$entry
+         data.idm$to[ind.cens.exit] <- 13
+         data.idm$exit[ind.cens.exit] <- data.idm$cens.time[ind.cens.exit]
+         data.idm <- data.idm[!ind.cens.entry, ]
+     }
+
+     data.idm
+ }
```

For all exercises, the matrix of logical defining the possible transitions is

```
> tra.idm <- matrix(FALSE, ncol = 3, nrow = 3)
> tra.idm[1, 2:3] <- TRUE
> tra.idm[2, 3] <- TRUE
```

a) Complete case

```
> data.comp <- simul.chap2.exo9(300, 0.5, 0.9, 0.8,
+                               cens = "none")
> require(mvna)
> fit.comp <- mvna(data.comp, c("0", "1", "2"), tra.idm, NULL)
> xyplot(fit.comp)
```

b) Random right-censoring

```
> data.random <- simul.chap2.exo9(300, 0.5, 0.9, 0.8,
+                                 cens = "random",
+                                 cens.param = list(first = c(0, 3)))
> fit.random <- mvna(data.random, c("0", "1", "2"), tra.idm, "13")
> xyplot(fit.random)
```

c) State-dependent censoring

```
> data.dep <- simul.chap2.exo9(300, 0.5, 0.9, 0.8,
+                              cens = "state",
+                              cens.param = list(first = c(0, 3),
+                              second = c(1.5, 4)))
> fit.dep <- mvna(data.dep, c("0", "1", "2"), tra.idm, "13")
> xyplot(fit.random)
```

d) E.g., Truncation and independent censoring

```
> data.random.trun <- simul.chap2.exo9(300, 0.5, 0.9, 0.8,
+                                      cens = "random",
+                                      cens.param = list(first = c(0, 3)),
+                                      trun = list(yes = TRUE,
+                                      shape = 1, scale = 1/3))
> fit.random.trun <- mvna(data.random.trun, c("0", "1", "2"), tra.idm, "13")
> xyplot(fit.random.trun)
```

Exercise 10

Show that a competing risks process is Markov.

Proof. As the competing risks model has only one transient state, and $\mathrm{P}(X_0 = 0) = 1$, it fulfils the Markov property. □

An illness-death model without recovery is Markov if the transition from the disease state to the state death does not depend on the duration since entry into the disease state.