Example 4 using R

Drawing time to event from hazards with time-dependent covariates

David Garibay, M.P.P.* Hawre Jalal, MD, Ph.D.†
Fernando Alarid-Escudero, Ph.D.‡§

Code function

This document presents the code corresponding to the fourth example presented in the "A Fast Nonparametric Sampling (NPS) Method for Time-to-Event in Individual-Level Simulation Models." manuscript, all of them using R.

```
# 01 Initial Setup ------------------------------------------------------

## 01.01 Clean environment ----------------------------------------------
remove(list = ls())

#* Refresh environment memory
gc()

# 01.02 Load libraries -------------------------------------------------
library(dplyr)
library(ggplot2)
library(tidyr)
library(tibble)
library(data.table)
library(flexsurv)
library(LambertW)
library(reshape2)
library(microbenchmark)
```

*School of Epidemiology and Public Health, Faculty of Medicine, University of Ottawa, Ottawa, ON, CA.
†School of Epidemiology and Public Health, Faculty of Medicine, University of Ottawa, Ottawa, ON, CA.
‡Department of Health Policy, Stanford University School of Medicine, Stanford, CA, USA.
§Center for Health Policy, Freeman Spogli Institute, Stanford University, Stanford, CA, USA.

```r
# Load function to implement multivariate categorical sampling
source(file = "../R/nps_nhppp.R")
```

```r
# 02 Define general parameters ------------------------------------------

# Parameters for time-varying covariates
alpha_0 <- 0
alpha_1 <- 1
# When beta <- 0 the time-varying covariate is deactivated
beta    <- log(1.02)

# Define a time range
time_var_cov <- seq(0, 100)

# Sample size
n_samp <- 1e6

#* Number of iterations for microbenchmarking in time-dependent covariates
#* examples
n_iter_time_var_cov <- 100

# Seed for reproducibility in random number generation
n_seed <- 10242022
```

```r
# 03 Define required functions ------------------------------------------

## Function to apply the time-varying covariate to a baseline hazard
compute_time_varying_hazard_linear_2 <- function(hazard0,
                                                 alpha_0,
                                                 alpha_1,
                                                 beta,
                                                 time_var_cov){

  # With this function we take the diagonal values of the matrix
  hazard <- hazard0*exp(beta*(alpha_0 + alpha_1*time_var_cov))

  # #* With this specification we obtain the full matrix and then we need to
  # #* make a subset considering a certain set of time-varying covariates
  # #* - DEACTIVATED
  # hazard <- hazard0 %*% exp(beta*(alpha_0 + alpha_1*time_var_cov))
```

```r
  return(hazard)
}


#* Function to get time to events from exponential baseliine hazard
#* using analytic formula following Austin's 2012 equations:
#* - Austin, P. C. (2012).
#*   Generating survival times to simulate Cox proportional hazards models
#*   with time-varying covariates. Statistics in Medicine, 31(29), 3946-3958.
#*   https://doi.org/10.1002/sim.5452
inv_exp_time_ <- function(n_samp, rate, alpha_0, alpha_1, beta) {

  v_unif <- runif(n = n_samp)

  exp_time <- (1/(beta*alpha_1))*log(1 +
                                       ((-alpha_1*beta*log(v_unif))/
                                          (rate*exp(alpha_0*beta)))
  )

  return(exp_time)
}

#* Function to get time to events from Gompertz baseliine hazard
#* using analytic formula following Austin's 2012 equations
inv_gomp_time_ <- function(n_samp,
                           shape,
                           rate,
                           alpha_0,
                           alpha_1,
                           beta) {

  v_unif <- runif(n = n_samp)

  gomp_time <- (1/((beta*alpha_1) + shape))*
    log(1 + (((beta*alpha_1) + shape)*(-log(v_unif)))/
          (rate*exp(alpha_0*beta)))

  return(gomp_time)
}

#* Function to get time to events from Gompertz baseliine hazard
#* using analytic formula following Ngwa, et al.'s equations:
```

```
#* - Ngwa, J. S., Cabral, H. J., Cheng, D. M., Gagnon, D. R., LaValley,
#*   M. P., & Cupples, L. A. (2022). Generating survival times with
#*   time-varying covariates using the Lambert W Function. Communications in
#*   Statistics: Simulation and Computation, 51(1), 135-153.
#    https://doi.org/10.1080/03610918.2019.1648822
inv_weibull_time <- function(n_samp,
                             shape,
                             scale,
                             alpha_0,
                             alpha_1,
                             beta) {

  v_unif <- runif(n = n_samp)

  weibull_time <- (1/(beta*alpha_1*(1/shape)))*LambertW::W(
    beta*(alpha_1*(1/shape))*(
      -log(v_unif)/scale*exp(beta*alpha_0)
    )^(1/shape)
  )

  return(weibull_time)
}
```

```
# 04 Draw time to events ------------------------------------------------

# Add seed for reproducibility
set.seed(n_seed)

## 04.01 Exponential baseline hazard ------------------------------------

# Define general parameters for the Exponential baseline hazard
rate    <- 0.1

# Obtain the exponential baseline hazard
v_exp_hazard0 <- matrix(data = flexsurv::hexp(x = 0:100, rate = rate),
                        ncol = 1)

#* Compute the hazard after adding a proportional hazards approach using a
#* time-varying covariate
hazard <- compute_time_varying_hazard_linear_2(
  hazard0      = v_exp_hazard0,
  alpha_0      = alpha_0,
```

```r
  alpha_1      = alpha_1,
  beta         = beta,
  time_var_cov = time_var_cov)


df_hazard_long <- reshape2::melt(data       = hazard,
                                 varnames   = c("Time", "Covariate"),
                                 value.name = "h(t)")

dt_hazard_long <- data.table::as.data.table(df_hazard_long)

#* Transform hazards `h(t)` to instantaneous probabilities `f(t)`
#* # H(t) - Cumulative hazard
dt_hazard_long[, H := cumsum(`h(t)`)]
# F(t) - Cumulative probability
dt_hazard_long[, `F` := 1 - exp(-H)]
# f(t) - Instantaneous probability
dt_hazard_long[,  f := c(`F`[1], diff(`F`))]

#* Sample times to event considering the previously defined instantaneous
#* probabilities
v_time_to_event_random_path <- sample(x    = time_var_cov,
                                      size = n_samp,
                                      prob = dt_hazard_long$f,
                                      replace = TRUE)

# Add continous time approximation
v_time_to_event_random_path <-  v_time_to_event_random_path +
  runif(n = length(v_time_to_event_random_path))


# Obtain times to event following analytical formula
v_exp_time <- inv_exp_time_(n_samp  = n_samp,
                            rate    = rate,
                            alpha_0 = alpha_0,
                            alpha_1 = alpha_1,
                            beta    = beta)

# Compare mean time to event of
## Analytical formula sample
ev_exp_time_af <- mean(v_exp_time)
## NPS method
```

```r
ev_exp_time_nps <- mean(v_time_to_event_random_path)

# Measure mean execution time
l_mbench_tvar_exp <- microbenchmark::microbenchmark(
  sample(x = 0:100,
         size = 1e6,
         prob = dt_hazard_long$f,
         replace = TRUE),
  times = n_iter_time_var_cov,
  unit = "ms")

## 04.02 Gompertz baseline hazard ----------------------------------------

# Define general parameters for the Weibull baseline hazard
shape <- 0.1
rate <- 0.001

# Obtain the Gompertz baseline hazard
v_gomp_hazard0 <- flexsurv::hgompertz(x = 0:100, shape = shape, rate = rate)

#* Compute the hazard after adding a proportional hazards approach using a
#* time-varying covariate
gomp_hazard <- compute_time_varying_hazard_linear_2(
  hazard0      = v_gomp_hazard0,
  alpha_0      = alpha_0,
  alpha_1      = alpha_1,
  beta         = beta,
  time_var_cov = time_var_cov)

df_gomp_hazard_long <- reshape2::melt(data       = gomp_hazard,
                                      varnames   = c("Time", "Covariate"),
                                      value.name = "h(t)")

dt_gomp_hazard_long <- as.data.table(df_gomp_hazard_long)

#* Transform hazards `h(t)` to instantaneous probabilities `f(t)`
dt_gomp_hazard_long[, H := cumsum(`h(t)`)]
dt_gomp_hazard_long[, `F` := 1 - exp(-H)]
dt_gomp_hazard_long[, f := c(`F`[1], diff(`F`))]

# Sample times to event considering the previously defined instantaneous
#probabilities
```

```r
v_time_to_event_gompertz <- sample(x = time_var_cov,
                                   size = n_samp,
                                   prob = dt_gomp_hazard_long$f,
                                   replace = TRUE)

# Add continous time approximation
v_time_to_event_gompertz <-  v_time_to_event_gompertz +
  runif(n = length(v_time_to_event_gompertz))

# Obtain times to event following analytical formula
v_gomp_time <- inv_gomp_time_(n_samp  = n_samp,
                              shape   = shape,
                              rate    = rate,
                              alpha_0 = alpha_0,
                              alpha_1 = alpha_1,
                              beta    = beta)


# Compare expected time to event of
## Analytical formula sample
ev_gomp_time_af <- mean(v_gomp_time)
## Proposed approach
ev_gomp_time_nps <- mean(v_time_to_event_gompertz)

# Measure mean execution time
l_mbench_tvar_gomp <- microbenchmark(
  sample(x = 0:100,
         size = 1e6,
         prob = dt_gomp_hazard_long$f,
         replace = T) +
    runif(n = length(v_time_to_event_gompertz)),
  times = n_iter_time_var_cov,
  unit = "ms")


## 04.03 Weibull baseline hazard ------------------------------------------

# Define general parameters for the Weibull baseline hazard
n_shape_weib = 2
n_scale_weib = 0.01

# Obtain the Weibull (proportional hazards) baseline hazard
```

```r
v_weibull_hazard0 <- matrix(flexsurv::hweibullPH(x     = time_var_cov,
                                                 shape = n_shape_weib,
                                                 scale = n_scale_weib))

#* Compute the hazard after adding a proportional hazards approach using a
#* time-varying covariate
weibull_hazard <- compute_time_varying_hazard_linear_2(
  hazard0      = v_weibull_hazard0,
  alpha_0      = alpha_0,
  alpha_1      = alpha_1,
  beta         = beta,
  time_var_cov = time_var_cov)

df_weibull_hazard_long <- reshape2::melt(data       = weibull_hazard,
                                         varnames   = c("Time", "Covariate"),
                                         value.name = "h(t)")

dt_weibull_hazard_long <- as.data.table(df_weibull_hazard_long)

# Sample time to events for random path
dt_weibull_hazard_long[, H := cumsum(`h(t)`)]
dt_weibull_hazard_long[, `F` := 1 - exp(-H)]
dt_weibull_hazard_long[, f := c(`F`[1], diff(`F`))]

# Sample times to event considering the previously defined instantaneous
# probabilities
v_time_to_event_weibull <- sample(x       = time_var_cov,
                                  size    = n_samp,
                                  prob    = dt_weibull_hazard_long$f,
                                  replace = TRUE)

# Add continous time approximation
v_time_to_event_weibull <-  v_time_to_event_weibull +
  runif(n = length(v_time_to_event_weibull))

# Obtain times to event following analytical formula
v_weibull_time <- inv_weibull_time(n_samp  = n_samp,
                                   shape   = n_shape_weib,
                                   scale   = n_scale_weib,
                                   alpha_0 = alpha_0,
                                   alpha_1 = alpha_1,
                                   beta    = beta)
```

```r
# Compare expected time to event of
## Analytical formula sample
ev_weibull_time_af <- mean(v_weibull_time)
## Proposed approach
ev_weibull_time_nps <- mean(v_time_to_event_weibull)

# Measure mean execution time
l_mbench_tvar_weib <- microbenchmark::microbenchmark(
  sample(x = 0:100,
         size = 1e6,
         prob = dt_weibull_hazard_long$f,
         replace = T) + runif(n = length(v_time_to_event_weibull)),
  times = n_iter_time_var_cov,
  unit = "ms")

# Remove seed
set.seed(NULL)
```