Example 5 using R

Drawing time to event from hazards with time-dependent covariates following random paths

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 fifth 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)
```

*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
library(microbenchmark)

# 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(1.005)


# Define parameters for the Weibull baseline hazard
n_weib_shape <- 1.3
n_weib_scale <- 30.1

n_ind    <-  1000 # Number of simulated individuals
n_cycles <-  100  # Number of cycles
ourDrift <-  0.5

#* 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 ------------------------------------------

# Define random path function
create_time_varying_covariate <- function(n_ind    = 100,
                                          n_cycles = 100,
                                          ourDrift = 0.005){

  m_random_paths <- matrix(nrow = n_ind, ncol = n_cycles)
  m_random_paths[, 1] <- 1

  for (cycle in 2:n_cycles) {

    v_next_step = rnorm(n = n_ind, mean = 0, sd = ourDrift)
```

```
    m_random_paths[, cycle] <- round(pmax(m_random_paths[, cycle - 1] +
                                            v_next_step, 0))
  }

  dtb_paths_individuals <- as.data.table(
    reshape2::melt(data       = m_random_paths,
                   varnames   = c("id", "Time"),
                   value.name = "Covariate"))

  setorder(dtb_paths_individuals, id, Time)

  return(dtb_paths_individuals)
}

## Function to apply the time-varying covariate to a baseline hazard
compute_time_varying_hazard_linear_3 <- function(hazard0,
                                                 alpha_0,
                                                 alpha_1,
                                                 beta,
                                                 time_var_cov){

  # This specification gets the full matrix of values as output
  hazard <- hazard0 %*% exp(beta*(alpha_0 + alpha_1*time_var_cov))

  return(hazard)
}
```

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

# Set seed for reproducibility
set.seed(n_seed)

# Sample from Weibull baseline hazard (h_0)
hazard0 <- matrix(flexsurv::hweibull(x     = 1:n_cycles,
                                     shape = n_weib_shape,
                                     scale = n_weib_scale),
                  ncol = 1)


# Compute values of the hazard based on a range of covariate values
weibull_hazard <- compute_time_varying_hazard_linear_3(
  hazard0      = hazard0,
```

```
  alpha_0      = alpha_0,
  alpha_1      = alpha_1,
  beta         = beta,
  time_var_cov = seq(0:n_cycles))

# Convert to long format
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)

# Correct covariate id
dt_weibull_hazard_long[, Covariate := Covariate - 1]

#* Set key for efficient binary search
#* Check `vignette("datatable-keys-fast-subset") `
setkey(dt_weibull_hazard_long, Time, Covariate)

# Create time varying covariate, y_i(t)
dtb_paths_individuals <- create_time_varying_covariate(n_ind    = n_ind,
                                                       n_cycles = n_cycles,
                                                       ourDrift = ourDrift)




# Obtain time-dependent hazards from indvidual-specific random paths
dtb_paths_individuals[, `h(t)` := dt_weibull_hazard_long[
  .(dtb_paths_individuals$Time, dtb_paths_individuals$Covariate),  `h(t)`]]


# Steps to get time-specific probability of event occurrence
# H(t) - Cumulative hazard
dtb_paths_individuals[, H := cumsum(`h(t)`),  by = id]
# F(t) - Cumulative probability
dtb_paths_individuals[, `F` := 1 - exp(-H)]
# f(t) - Instantaneous probability
dtb_paths_individuals[, f := c(`F`[1], diff(`F`)), by = id]

# Generate data set to sample time to event
dt_paths_individuals_wide <- data.table::dcast(data = dtb_paths_individuals,
```

```
                                                value.var = "f",
                                                formula = id ~ Time)

# Generate last cycle to sum probability up to 1
dt_paths_individuals_wide[
  , `101` := 1 - dtb_paths_individuals[Time == 100, `F`]]

# Sample time to event for all individuals
out_nps <- nps_nhppp(
  m_probs = as.matrix(dt_paths_individuals_wide[, `1`:`101`]),
  v_categories = seq(0, 100),
  correction = "none")

# Measure mean execution time
l_mbench_random_path <- microbenchmark::microbenchmark(
  nps_nhppp(m_probs    = as.matrix(dt_paths_individuals_wide[, 2:102]),
            correction = "uniform"),
  times = n_iter_time_var_cov,
  unit = "ms")

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