Definition of the NPS function in R

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

Code function

This document presents an R implementation of the multivariate categorical sampling function mentioned in the "A Fast Nonparametric Sampling (NPS) Method for Time-to-Event in Individual-Level Simulation Models." manuscript, and used in the R examples provided in the Appendix.

```r
#' Nonparametric sampling of time to events from a discrete nonhomogeneous
#' Poisson point process
#'
#' \code{nps_nhppp} samples states for multiple individuals simultaneously.
#'
#' The elements of each row of the matrix `m_probs` should sum up to 1. In
#' case this does not happens, all the rows where this condition is not met
#' will be normalized.
#'
#' @param m_probs matrix with probabilities for each category. Each row
#' represents one individual and each column an specific category.
#' @param v_categories An optional argument. It is a vector containing the
#' name of the categories to sample from. The length of this vector must be
#' equal to the number of columns of `m_probs`.
#' @return A vector filled with sampled categories.
#' @export
nps_nhppp <- function(m_probs,
                      v_categories = NULL,
```

[*]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
                     correction = c("none", "uniform")) {

if (!is.numeric(m_probs)) {
  stop("`m_probs` must be a matrix filled with numeric elements")
}

if (!isTRUE(all.equal(target = rep(1, nrow(m_probs)),
                      current = as.numeric(rowSums(m_probs))))) {

  #* Find rows where the sum of their elements is not equal to 1.
  #* We use a very small value to check that they are equal to 1 since
  #* == and != sometimes do not work as desired. The value 1.5e-8
  #* was taken as the same level of tolerance as the `all.equal` function.
  v_rows_to_norm <- which(abs(1 - rowSums(m_probs)) > 1.5e-8)

  warning(
    "The rows: ",
    paste(as.character(v_rows_to_norm), collapse = ", "),
    " in `m_probs` do not sum up to 1. The values within these rows will be
    normalized and then used in the sampling process. In case this behaviour
    is not desired modify the content of `m_probs`.")

  # Normalize rows
  if (length(v_rows_to_norm) == 1) {
    m_probs[v_rows_to_norm, ] <- m_probs[v_rows_to_norm, ]/
      sum(m_probs[v_rows_to_norm, ])
  }
  if (length(v_rows_to_norm) > 1) {
    m_probs[v_rows_to_norm, ] <- m_probs[v_rows_to_norm, ]/
      rowSums(m_probs[v_rows_to_norm, ])
  }
}

# Number of categories to sample from
n_cat <- ncol(m_probs)

if (is.null(v_categories)) {
  #' Generate the numeric categories based on the number of columns of
  #' `m_probs`
  v_categories <- seq(0, (n_cat - 1))
}
```

```r
  correction <- match.arg(correction)

  # Number of time to events to draw
  n_samp <- nrow(m_probs)

  v_unif <- runif(n_samp, min = 0, max = 1)
  v_sum_p <- matrixStats::rowCumsums(m_probs)
  v_time_to_event <- v_categories[max.col(v_sum_p >= v_unif,
                                            ties.method = "first")]

  if (correction == "uniform") {
    v_time_to_event <- v_time_to_event + runif(n_samp, min = 0, max = 1)
  }
  return(v_time_to_event)

}
```