A simple discrete event simulation model of a cancer's natural history

TA Trikalinos, Y Sereda, S Chrysanthopoulou, F Alarid-Escudero

2024-10-17

The simulation world

We will model a cancer's natural history in a population. We index people by $k \in [K] := \{1, \dots, K\}$. The following assumptions fix a simulation world.

1. All types of events can be modeleld with nonhomogeneous Poisson point processes (NHPPPs).
2. Persons are alive and cancer free at 40 years of age. No person will live past 110 years. All people can die from causes other than the cancer of interest (hereafter, *death from other causes*). Write $\rho_k(t)$ for the corresponding intensity function.
3. Some people may be exposed to an environmental toxin, with exposure that varies over time. Write $\xi_k(t) \geq 0$ for the exposure function. Positive values mean that a person is exposed at that particular instance. (The never exposed have zeros throughout their life.) We will assume that the exposure to said toxin is a risk factor only for cancer emergence, and that the toxin has no cumulative effects – only the instantaneous exposure levels matter. Write δ_k for the effect of the toxin instantaneous exposure on developing cancer.
4. Some persons will develop a preclinical cancer with a time-varying intensity function $\lambda_k(t) = \underbrace{\lambda_{k0}(t)}_{\text{non-risk factor part}} + \underbrace{\delta_k \, \xi_k(t)}_{\text{risk factor part}}$.
5. Some preclinical cancers may progress to clinical cancer with an intensity function $\mu_k(t)$, at which point they are considered diagnosed.
6. Once people develop preclinical cancer they can die from cancer with intensity function $\nu_k(t)$. The cancer death rate does not explicitly depend on whether the cancer has been diagnosed or not. Thus, we have two competing causes of death: death due to cancer and due to other causes.

Setup

Load `nhppp` and `data.table`. (If you prefer to not use `data.table`, you should be able to implement this example in base `R` with little trouble.)

```
library(data.table)
library(nhppp)
```

Simulation model

We now fix the mathematical description of the model. We will simulate $K = 10^4$ males and females (with equal probability) from the 2015 birth cohort.

```
pop <- data.table(
  id = 1:K,
  birth_cohort = 2015,
  spawn_age = 40,
  max_simulation_age = 110,
  sex = sample(c("male", "female"), K, replace = TRUE)
)

## It would make sense to execute the commented-out code now.
## It generates model parameters used in later stages.
## For expository clarity, we generate each parameter when it is introduced
# pop[, `:=`(
#    param_cancer_emergence_shape = runif(.N, 7, 9),
#    param_cancer_emergence_scale = rnorm(.N, 150, 20),
#    param_toxin_exposure_diff = pmax(0.005, rnorm(.N, 0.01, 0.005)),
#    param_cancer_death_intercept := rnorm(.N, -2, 0.5),
#    param_cancer_death_slope := runif(n= .N, min = 0, max = 0.003),
#    param_clinical_cancer_dx_rate := runif(.N, 0.20, 0.27)
#  )]
```

Death from other causes

The death from other causes $\rho_k(t)$ depends on the age (t, measured in years), sex (male or female), and birth year of person k. Function ρ is is a piecewise constant over each year of age. It is a `regular' step function (all steps have the same length of one year).

If the cancer is not a major cause of death, then the intensity function for all cause deaths is a good approximation for the intensity function for death from other causes. The internal dataset `annual_mortality_rates_2015` has all cause mortality data for the 2015 birth cohort. It has the values of the piecewise constant ρ per birth cohort, sex, and age. Here is a peek at some columns.

```
annual_mortality_rates_2015[
  sex %in% c("male", "female"),
  c(1:5, 111:113)
]
#> Key: <birth_cohort, sex>
#>    birth_cohort    sex    age_0    age_1    age_2  age_108  age_109 age_110+
#>           <int> <fctr>    <num>    <num>    <num>    <num>    <num>    <num>
```

```
#>    1:         2015 female 0.005386 0.000350 0.000228 0.559371 0.541174 0.587413
#>    2:         2015   male 0.006404 0.000452 0.000277 0.511677 0.671391 0.386100
```

When we have a *step* (piecewise constant) intensity function over *regular* time intervals (here, all one year long), we can use `nhppp`'s `vdraw_sc_step_regular()` function. We need to specify the following:

1. Pass the intensities as a matrix (argument `lambda_matrix`); the number of columns in the matrix are the number of time intervals in the step function.

```
rhos <- annual_mortality_rates_2015[
  pop,
  on = c("birth_cohort", "sex")
]
setindex(rhos, "id")
rho_matrix <- as.matrix(rhos[, c(paste0("age_", 0:109), "age_110+"),
  with = FALSE
])

rm(list = "rhos") # cleanup
```

2. Give information about how long each time step is, by specifying the age bounds `rate_matrix_t_min` and `rate_matrix_t_max` over which the intensity matrix applies;
3. Optionally, if we want to sample times in a sub-interval of `(rate_matrix_t_min, rate_matrix_t_max]`, we can specify even narrower bounds, `t_min` and `t_max`. If you omit `t_min` or `t_max`, the software uses `rate_matrix_t_min` or `rate_matrix_t_max`, respectively, to specify the sampling interval.
4. Because no person lives beyond the maximum simulation age of 110 years, we need to force the simulation of at least one death event over the simulation interval. This means that we are sampling from a zero-truncated NHPPP. Setting the option `atleast1` to `TRUE` achieves this.
5. We only need to sample the earliest event from this NHPPP. So we set the `atmost1` option to `TRUE`.

```
pop[
  ,
  age_dead_from_other_causes :=
    nhppp::vdraw_sc_step_regular(
      lambda_matrix = rho_matrix,
      rate_matrix_t_min = 0,
      rate_matrix_t_max = 110,
      t_min = pop$spawn_age, # 40
      t_max = pop$max_simulation_age, # 110
      atmost1 = TRUE,
      atleast1 = TRUE
    )
]
```

Environmental toxin exposure histories

We will generate environmental exposure histories with a phenomenological model. We will assume that

1. People may be exposed to the environmental toxin with probability $p_{start} = 0.20$.
2. For those who will be exposed, the start age of exposure is $t_{k0} \sim U(12, 35)$, provided that they are still alive.
3. Among those who are exposed, the probability that their exposure will eventually stop is $p_{stop} = 0.60$.
4. In the pertinent subgroup of persons, the duration of the exposure is $d_k \sim U(1, 35)$, if they are still alive.
5. For people with at least some exposure to the toxin, for all times in the exposure window $(t_{k0}, t_{k1}]$ the exposure levels are $\xi_k(t) = \Xi_k \cdot \left(\frac{1}{2} + \frac{1}{4}\left(\cos(0.5t) + \cos(0.45t)\right)\right)$, where t is a person's age and the amplitude (maximum exposure) Ξ_k has model $\Xi_k \sim U(0.2, 1)$. Otherwise, $\xi_k(t) = 0$.

We now add the per-person parameters for exposure histories in the population data.table. For people who will never be exposed we set Ξ_k to zero, and their exposure start and stop ages at the `max_simuation_age`. This avoids `if ... else` statements, and is still pretty fast. If you run a massive model, though, you may want to be smarter about it.

```
pop[, `:=`(
  exposure_start_age = max_simulation_age,
  exposure_stop_age = max_simulation_age,
  maximum_exposure = 0
)][
  ,
  will_start_exposure := runif(.N) < 0.20
][
  will_start_exposure == TRUE,
  will_stop_exposure := runif(.N) < 0.60
][
  will_start_exposure == TRUE,
  exposure_start_age := pmin(runif(.N, 12, 35), age_dead_from_other_causes)
][
  will_stop_exposure == TRUE,
  exposure_stop_age := pmin(
    exposure_start_age + runif(.N, 1, 35),
    age_dead_from_other_causes
  )
][
  will_start_exposure == TRUE,
  maximum_exposure := runif(.N, 1 / 5, 1)
]
```

```
# cleanup
pop[, will_start_exposure := NULL][, will_stop_exposure := NULL]
```

We implement $\xi_k(t)$ as a function that is vectorized over all its arguments. The arguments `start_age`, `stop_age`, `max_exposure` correspond to the variables t_{k0}, t_{k1}, Ξ in the equation above.

```
xi <- toxin_exposure <- function(t, max_exposure, start_age, stop_age) {
  (start_age <= t) * (stop_age >= 1) * max_exposure * (1 / 2 + (cos(t / 2) + cos(0.9 * t /
        2)) / 4)
}
```

Emergence of pre-clinical cancer in unexposed and exposed intervals

Assume that the intensity function for cancer emergence in the absence of toxin exposure is

$$\lambda_{k0}(t) = \frac{shape_k}{scale_k}\left(\frac{t}{scale_k}\right)^{shape_k - 1},$$

where t is age in years.

This intensity function generates a Weibull point process (a special case of an NHPPP). The parameters $shape_k$ and $scale_k$ are assumed to vary across people according to the models $shape_k \sim U(7, 9)$ and $scale_k \sim N(150, 20)$, where $U(\cdot)$ and $N(\cdot)$ stand for uniform and normal distributions. We sample these values for each person in the population.

```
pop[, `:=`(
  param_cancer_emergence_shape = runif(.N, 7, 9),
  param_cancer_emergence_scale = rnorm(.N, 150, 20)
)]
```

Generating a Weibull point process is easy in `R` using the `stats::rweibull()` function. (This would take care of the cancer emergence times for people without toxin exposure, but not for people with toxin exposures.) Accounting for toxin exposure histories, the intensity function for cancer emergence is $\lambda_k(t) = l_{k0}(t) + \delta_k \xi_k(t)$.

We will assume that the toxin exposure effect δ_k is distributed as $\delta_k \sim N_+\left(2, 0.5\right)$, where $N_+(\cdot)$ is a slab and smear normal distribution.

```
pop[, param_toxin_exposure_diff := pmax(0, rnorm(.N, 0.01, 0.005))]
```

We need to sample from an NHPPP with an intensity function that varies over time as per $\lambda_k(t)$. We will use `nhppp`'s `vdraw_intensity()` function, which needs

1. The intensity function (argument `lambda`) in a vectorized form, so that age t is the only needed argument and all other arguments are set by default.
2. A majorizer piecewise constant function, which will be specified as a matrix `lambda_maj_matrix`.

3. The `rate_matrix_t_min`, `rate_matrix_t_max` arguments that specify the time bounds for the matrix `lambda_maj_matrix`.
4. `t_min`, and `t_max` arguments, for the subinterval over whchi we will sample times. Here, `t_min = 40` – the spawn age in the simulation, and `t_max` is the `age_dead_from_other_causes`.

Let's implement the above.

The *intensity function* is specified as follows. Observe that it is in vectorized form.

```
lambda <- function(t, P = pop, ...) {
  # non-risk factor part: shape / scale * (t/scale)^(shape - 1)
  (P$param_cancer_emergence_shape / P$param_cancer_emergence_scale) *
    (t / P$param_cancer_emergence_scale)^(P$param_cancer_emergence_shape - 1) +
    # risk factor (toxin exposure) part: delta_k * xi(t)
    P$param_toxin_exposure_diff *
      xi(
        t = t,
        max_exposure = P$maximum_exposure,
        start_age = P$exposure_start_age,
        stop_age = P$exposure_stop_age
      )
}
```

We also need a piecewise constant *majorizer* function $\lambda_*(t)$. We say that $\lambda_*(t)$ majorizes $\lambda(t)$ if $\lambda_*(t) \geq \lambda(t)$ for all t of interest. The function expects a *regular step* majorizer function. We will create such a function with $M = 10$ equally-spaced intervals over the whole simulation window. First, generate the endpoints of the M intervals. This will be a matrix where the rows correspond to persons and the columns to the $M + 1 = 11$ interval bounds.

```
# define interval bounds for the step function, one row per person
M <- 10
time_breaks <- matrix(
  data = rep(x = seq(from = 40, to = 110, length.out = M + 1), each = K),
  byrow = FALSE,
  nrow = K
)

time_breaks[1:3, ]
#>      [,1] [,2] [,3] [,4] [,5] [,6] [,7] [,8] [,9] [,10] [,11]
#> [1,]   40   47   54   61   68   75   82   89   96   103   110
#> [2,]   40   47   54   61   68   75   82   89   96   103   110
#> [3,]   40   47   54   61   68   75   82   89   96   103   110
```

… and now generate the majorizer matrix using `nhppp`'s `get_step_majorizer()` function. (The paper in the Bibliography explains how this function works.)

```
lambda_star <- nhppp::get_step_majorizer(
  fun = lambda,
  breaks = time_breaks,
  is_monotone = FALSE,
  K = 1.9 / 4 # This is the maximum slope of xi() -- which you get with some calculus
)

lambda_star[1:3, ]
#>          [,1]     [,2]     [,3]     [,4]     [,5]     [,6]     [,7]     [,8]
#> [1,] 1.662507 1.662520 1.662549 1.662608 1.662719 1.662918 1.663257 1.663810
#> [2,] 1.662539 1.662591 1.662691 1.662869 1.663168 1.663648 1.664388 1.665489
#> [3,] 1.662542 1.662609 1.662752 1.663035 1.663554 1.664454 1.665944 1.668315
#>          [,9]    [,10]
#> [1,] 1.664681 1.666012
#> [2,] 1.667083 1.669331
#> [3,] 1.671963 1.677415
```

And now, we can sample the cancer generation times, and create a variable to identify patients with cancer.

```
pop[
  ,
  age_cancer_emergence := nhppp::vdraw_intensity(
    lambda = lambda,
    lambda_maj_matrix = lambda_star,
    rate_matrix_t_min = 40,
    rate_matrix_t_max = 110,
    t_min = pop$spawn_age,
    t_max = pmin(pop$age_dead_from_other_causes, 110, na.rm = TRUE),
    atmost1 = TRUE
  )
][
  ,
  with_cancer := !is.na(age_cancer_emergence),
]
```

Dying from cancer

People with preclinical cancer may die from cancer causes. We will assume that the intensity from cancer deaths is loglinear in time, that is

$\nu_k = e^{\alpha_k + \beta_k t}$,

with parameters $\alpha_k \sim N(-3, 0.2)$ and β_k $U(0, 0.003)$.

```
pop[, param_cancer_death_intercept := rnorm(.N, -3, 0.2)]
pop[, param_cancer_death_slope := runif(.N, 0, 0.003)]
```

We could use the `vdraw_intensity()` function again, since we already know the intensity ν_k and we can easily create a majorizer function for it, as we did when we generated cancer emergence times. This would be plenty fast for our small simulation with $K = 10^4$ and requires no additional mathematics.

We can sample even faster if we can analytically obtain the cumulative intensity function $N_k(t) = \int_0^t \nu_k(s)\, \mathrm{d}s$, and its inverse $N_k^{-1}(z)$. (The inverse function recovers t from the value of $N_k(t)$: $t = N_k^{-1}\left(N_k(t)\right)$). This sampling is done with `nhppp`'s `vdraw_cumulative_intensity()` function.

A bit of calculus can yield $N_k(t) = \frac{1}{\beta_k}(e^{\alpha_k + \beta_k t} - e^{\alpha_k})$, which we can implement in vectorized form and with default parameters already set:

```
Nu <- function(t, Lambda_args = list(population), ...) {
  P <- Lambda_args$population
  (
    exp(P$param_cancer_death_intercept + P$param_cancer_death_slope * t) -
      exp(P$param_cancer_death_intercept)
  ) / P$param_cancer_death_slope
}
```

The inverse $N_k^{-1}(\cdot)$ is

$N_k^{-1}(z) = ((_k z + e^_k) - _k)/_k $

```
Nu_inv <- function(z, Lambda_inv_args = list(population), ...) {
  P <- Lambda_inv_args$population
  (
    log(P$param_cancer_death_slope * z +
      exp(P$param_cancer_death_intercept)) -
      P$param_cancer_death_intercept
  ) / P$param_cancer_death_slope
}
```

Then, we can sample the times to cancer death.

```
args_list <- list(population = pop[!is.na(age_cancer_emergence), ])
pop[
  !is.na(age_cancer_emergence),
  age_dead_from_cancer_causes := nhppp::vdraw_cumulative_intensity(
    Lambda = Nu,
    Lambda_args = args_list,
    Lambda_inv = Nu_inv,
    Lambda_inv_args = args_list,
```

```
    t_min = pop[!is.na(age_cancer_emergence), age_cancer_emergence],
    t_max = pop[!is.na(age_cancer_emergence), age_dead_from_other_causes],
    atmost1 = TRUE
  )
]

rm(list = "args_list") # cleanup
```

Dying from all causes

The age of death from all causes is the minimum of the ages across both causes of death.

```
pop[
  ,
  age_dead := pmin(age_dead_from_other_causes,
    age_dead_from_cancer_causes,
    na.rm = TRUE
  )
]
```

Clinical cancer diagnosis

Cancers first emerge in a pre-clinical stage. Some will be diagnosed as clinical cancers with intensity function μ_k. We will assume that clinical diagnosis has constant rate which is distributed according to the model

$$\mu_k(t) := \mu_k \sim U(0.20, 0.27),$$

where k indexes over people with cancer.

```
pop[
  !is.na(age_cancer_emergence),
  param_clinical_cancer_dx_rate := runif(.N, 0.20, 0.27)
]
```

Constant rates result in exponential times, which we can easily sample with the `stats::rexp()` function, as per the commented out code below.

```
### Using rexp()
tictoc::tic()
pop[
  !is.na(age_cancer_emergence),
  age_clinical_cancer_dx :=
    age_cancer_emergence +
```

```
    rexp(.N, rate = param_clinical_cancer_dx_rate)
]
pop[
  age_clinical_cancer_dx >= age_dead,
  age_clinical_cancer_dx := NA
]
tictoc::toc()
#> 0.001 sec elapsed
```

With `nhppp`, we can use the `vdraw_sc_step_regular()` function that samples from piecewise constant intensities. (A constant function over an interval is still a piecewise constant function – with a single piece.) The `nhppp` implementation will be only a bit slower – but it is worth showing.

```
tictoc::tic()
mu_mat <- as.matrix(pop[
  !is.na(age_cancer_emergence),
  param_clinical_cancer_dx_rate
])

pop[
  !is.na(age_cancer_emergence),
  age_clinical_cancer_dx := nhppp::vdraw_sc_step_regular(
    lambda_matrix = mu_mat,
    rate_matrix_t_min = pop[!is.na(age_cancer_emergence), age_cancer_emergence],
    rate_matrix_t_max = pop[!is.na(age_cancer_emergence), age_dead],
    atmost1 = TRUE
  )
]
tictoc::toc()
#> 0.001 sec elapsed
```

Some descriptives

```
# pop$age_cancer_emergence |> summary()
summary(pop)
#>        id         birth_cohort    spawn_age  max_simulation_age
#>  Min.   :    1   Min.   :2015   Min.   :40   Min.   :110
#>  1st Qu.: 2501   1st Qu.:2015   1st Qu.:40   1st Qu.:110
#>  Median : 5000   Median :2015   Median :40   Median :110
#>  Mean   : 5000   Mean   :2015   Mean   :40   Mean   :110
#>  3rd Qu.: 7500   3rd Qu.:2015   3rd Qu.:40   3rd Qu.:110
#>  Max.   :10000   Max.   :2015   Max.   :40   Max.   :110
#>
#>      sex           age_dead_from_other_causes exposure_start_age
```

```
#>  Length:10000       Min.   : 40.00            Min.   : 12.01
#>  Class :character   1st Qu.: 72.82            1st Qu.:110.00
#>  Mode  :character   Median : 82.54            Median :110.00
#>                     Mean   : 80.17            Mean   : 92.71
#>                     3rd Qu.: 89.55            3rd Qu.:110.00
#>                     Max.   :110.00            Max.   :110.00
#>
#>  exposure_stop_age maximum_exposure param_cancer_emergence_shape
#>  Min.   : 13.78    Min.   :0.0000   Min.   :7.000
#>  1st Qu.:110.00    1st Qu.:0.0000   1st Qu.:7.491
#>  Median :110.00    Median :0.0000   Median :7.987
#>  Mean   :101.92    Mean   :0.1221   Mean   :7.989
#>  3rd Qu.:110.00    3rd Qu.:0.0000   3rd Qu.:8.481
#>  Max.   :110.00    Max.   :0.9998   Max.   :9.000
#>
#>  param_cancer_emergence_scale param_toxin_exposure_diff age_cancer_emergence
#>  Min.   : 81.73               Min.   :0.000000          Min.   : 40.19
#>  1st Qu.:136.20               1st Qu.:0.006734          1st Qu.: 59.45
#>  Median :149.56               Median :0.010018          Median : 71.85
#>  Mean   :149.79               Mean   :0.010122          Mean   : 70.34
#>  3rd Qu.:163.17               3rd Qu.:0.013482          3rd Qu.: 81.85
#>  Max.   :231.62               Max.   :0.028532          Max.   :101.70
#>                                                         NA's   :9548
#>  with_cancer     param_cancer_death_intercept param_cancer_death_slope
#>  Mode :logical   Min.   :-3.733              Min.   :4.129e-07
#>  FALSE:9548      1st Qu.:-3.134              1st Qu.:7.549e-04
#>  TRUE :452       Median :-3.002              Median :1.518e-03
#>                  Mean   :-3.000              Mean   :1.501e-03
#>                  3rd Qu.:-2.866              3rd Qu.:2.239e-03
#>                  Max.   :-2.226              Max.   :2.999e-03
#>
#>  age_dead_from_cancer_causes    age_dead      param_clinical_cancer_dx_rate
#>  Min.   : 42.96              Min.   : 40.00   Min.   :0.200
#>  1st Qu.: 62.55              1st Qu.: 72.25   1st Qu.:0.217
#>  Median : 75.08              Median : 82.17   Median :0.235
#>  Mean   : 73.72              Mean   : 79.81   Mean   :0.235
#>  3rd Qu.: 85.19              3rd Qu.: 89.32   3rd Qu.:0.254
#>  Max.   :105.72              Max.   :110.00   Max.   :0.270
#>  NA's   :9749                                 NA's   :9548
#>  age_clinical_cancer_dx
#>  Min.   : 40.45
#>  1st Qu.: 60.03
#>  Median : 73.32
#>  Mean   : 71.21
#>  3rd Qu.: 81.37
#>  Max.   :100.73
#>  NA's   :9689
```

Bibliography

Trikalinos TA, Sereda Y. *nhppp: Simulating Nonhomogeneous Poisson Point Processes in R*. arXiv preprint arXiv:2402.00358. 2024 Feb 1.

Since the publication of the paper, the syntax and options of the `nhppp` package have evolved. To reproduce the code in the paper, you have to install the version of `nhppp` used in the paper. Alternatively, take a look at the vignettes, which are written to work with the current package.