Bias_NB

Amirhosein

2025-07-05

```
library("timeROC")
library("survival")
library("dcurves")
library("dplyr")
```

```
##
## Attaching package: 'dplyr'

## The following objects are masked from 'package:stats':
##
##     filter, lag

## The following objects are masked from 'package:base':
##
##     intersect, setdiff, setequal, union
```

```
library("survivalROC")
library("ggplot2")
```

```
sim_noninformative_censoring <- function(
  n           = 5000,    # sample size
  beta        = 1.5,   # log-HR for marker → event
  lambda      = 0.1,     # Weibull scale for events
  gamma       = 1.5,     # Weibull shape for events
  target_cens = 0.3,   # overall censoring proportion
  seed        = 123
) {
  set.seed(seed)

  # (a) simulate marker & true event-time
  marker    <- runif(n)
  eta       <- beta * marker
  u         <- runif(n)
  true_time <- (-log(u) / (lambda * exp(eta)))^(1/gamma)

  # (b) non-informative censoring
  med_t     <- median(true_time)
  base_rate <- -log(1 - target_cens) / med_t
  cens_t    <- rexp(n, rate = base_rate)

  # (c) observed time & status
  obs_time <- pmin(true_time, cens_t)
  status   <- as.integer(true_time <= cens_t)
```

```r
  data.frame(
    id         = seq_len(n),
    marker     = marker,
    true_time  = true_time,
    obs_time   = obs_time,
    status     = status
  )
}
```

```r
sim_informative_censoring <- function(
  n           = 5000,
  beta        = 1.5,      # log-HR marker → event
  lambda      = 0.1,      # Weibull scale
  gamma       = 1.5,      # Weibull shape
  target_cens = 0.3,
  gamma_cens  = 2,
  seed        = 123
)
{
  set.seed(seed)
  marker    <- runif(n)
  eta       <- beta * marker
  u         <- runif(n)
  true_time <- (-log(u) / (lambda * exp(eta)))^(1/gamma)
  unscaled  <- exp(gamma_cens * marker)


  find_s    <- function(s) {
    cens_t <- rexp(n, rate = s * unscaled)
    mean(true_time > cens_t) - target_cens
  }
  s_star     <- uniroot(find_s, c(1e-6, 100), tol=1e-5)$root


  cens_time  <- rexp(n, rate = s_star * unscaled)
  obs_time   <- pmin(true_time, cens_time)
  status     <- as.integer(true_time <= cens_time)


  data.frame(
             marker    = marker,
             true_time = true_time,
             obs_time  = obs_time,
             status    = status)
}
```

```r
compare_dca_methods <- function(sim, t0, cuts) {
  # 1) True net-benefit -------------------------------------------------------
  sim <- sim %>% mutate(status_true = as.integer(true_time <= t0))
  N <- nrow(sim)
  events_true <- sum(sim$status_true)
  nonevents <- N - events_true
  prev_true <- events_true / N
```

```r
nb_true <- sapply(cuts, function(z) {
  TP <- sum(sim$marker >= z & sim$status_true == 1)
  FP <- sum(sim$marker >= z & sim$status_true == 0)
  Se <- TP / events_true
  FPR <- FP / nonevents
  prev_true * Se - (1 - prev_true) * FPR * (z / (1 - z))
})

true_tbl <- tibble(threshold = cuts, net_benefit_true = nb_true)

# 2) Naïve DCA (KM-based) ---------------------------------------------------
dca_naive <- dca(Surv(obs_time, status) ~ marker, data = sim, time = t0, thresholds = cuts)
naive_tbl <- as_tibble(dca_naive) %>%
  filter(variable == "marker") %>%
  select(threshold, net_benefit) %>%
  rename(net_benefit_naive = net_benefit)

# 3) IPCW-corrected DCA with correct prevalence -----------------------------

# Fit Cox model for censoring distribution using covariate X (marker)
cox_censor <- coxph(Surv(obs_time, 1 - status) ~ marker, data = sim)

# Estimate S_c(Z_i | X_i) for each subject at their observed time Z_i
sc_fit <- survfit(cox_censor, newdata = sim)
S_c_Zi_Xi <- summary(sc_fit, times = sim$obs_time, extend = TRUE)$surv

# Define IPCW weights: w_i = 1 / S_c(Z_i | X_i)
w_i <- 1 / S_c_Zi_Xi

# Compute IPCW-adjusted prevalence:
prev_ipcw <- mean(w_i * (sim$obs_time <= t0 & sim$status == 1))


# Calculate IPCW-adjusted net benefit using SeSpPPVNPV for Se, FPR
nb_ipcw <- sapply(cuts, function(z) {
  res <- SeSpPPVNPV(
    cutpoint = z,
    T = sim$obs_time,
    delta = sim$status,
    marker = sim$marker,
    cause = 1,
    weighting = "cox",
    times = t0
  )
  Se <- res$TP[2]
  FPR <- res$FP[2]
  prev_ipcw * Se - (1 - prev_ipcw) * FPR * (z / (1 - z))
})

ipcw_tbl <- tibble(
  threshold = cuts,
  net_benefit_ipcw = nb_ipcw
)
```

```r
  # 4) Comparison table -------------------------------------------------------
  cmp <- true_tbl %>%
    inner_join(naive_tbl, by = "threshold") %>%
    inner_join(ipcw_tbl, by = "threshold") %>%
    mutate(
      diff_naive = abs(net_benefit_true - net_benefit_naive),
      diff_ipcw = abs(net_benefit_true - net_benefit_ipcw)
    )

  cat("Sum of biases (naïve):", sum(cmp$diff_naive), "\n")
  cat("Sum of biases (IPCW) :", sum(cmp$diff_ipcw), "\n")

  invisible(list(
    true = true_tbl,
    naive = naive_tbl,
    ipcw = ipcw_tbl,
    comparison = cmp
  ))
}
```

```r
cuts <- seq(0, 0.75, by = 0.01)
t0   <- 3
```

```r
sim_non  <- sim_noninformative_censoring(n = 5000, target_cens = 0.3)
results <- compare_dca_methods(sim_non, t0, cuts)
```

```
## Sum of biases (naïve): 0.162882
## Sum of biases (IPCW) : 0.1797458
```

```r
sim_inf  <- sim_informative_censoring(n = 5000, target_cens = 0.3)
results <- compare_dca_methods(sim_inf, t0, cuts)
```

```
## Sum of biases (naïve): 0.9896343
## Sum of biases (IPCW) : 0.3848641
```