Bayesian Causal Inference for Recurrent Events with Timing Misalignment

Yuqin

2025-08-15

This Markdown of the Bayesian framework was proposed by Oganisian et al. (2024) for causal inference with recurrent events subject to timing misalignment. We use a semiparametric Bayesian model to estimate the average causal effect of a time-varying treatment on the recurrent event rate, accounting for terminal events and censoring.

1. Setup

We configure Stan and set parallelization options for efficient MCMC workflow. We fix the random seed, enable parallel sampling, and set Hamiltonian Monte Carlo controls. Stan uses warmup internally and excludes warmup draws from the posterior sample used for inference, and all summaries below are computed from the post-warmup draws (as returned by `rstan::extract()`)

```
use_multicore <- TRUE   # FALSE for single core
if (use_multicore) {
  cores <- max(1, detectCores() - 2)
} else {
  cores <- 1
}
options(mc.cores = cores)
rstan_options(auto_write = TRUE)

B      <- 50
s_vec  <- c(3, 6, 9)
```

2. Data Preprocessing

This step prepares the longitudinal dataset for analysis, including normalization and handling of missing values.

The package expects a long panel with: unique subject id (id), discrete time index (k), treatment (A_k), recurrent count (Y_k), terminal event indicator (T_k), and optional covariates (here $L.1$, $L.2$).

History $lagY_k$ (default 0 at each subject's first interval) captures short-term dependence. In our design, lagged history terms are fully flexible: users may supply their own lag variables (e.g., lagYk, higher-order lags, or custom functions of past outcomes) directly in the dataset and reference them in the formulas, or omit them entirely. If no lag variable is provided but lag terms are specified in the formulas, the function automatically constructs the appropriate lag internally. This guarantees valid defaults while allowing investigators to encode any history structure they need.

Continuous covariates are standardized to improve sampler geometry and prior interpretability. Remove observations with missing values for variables used in the model formulas. K equals the number of unique time intervals and controls the length of each model's piecewise baseline.

```
load("data.Rdata")
df_fit <- df %>%
  filter(id %in% 1:100) %>%           #for subset
  arrange(id, k) %>%
  mutate(k_fac = as.integer(factor(k, levels = sort(unique(k))))) %>%
  group_by(id) %>%
  mutate(
    lagYk = if ("lagYk" %in% names(.)) replace_na(lagYk, 0) else lag(Yk, default = 0)
  ) %>%
  ungroup() %>%
  drop_na(Tk, Yk, Ak, L.1, L.2) %>%
  mutate(
    L.1 = as.numeric(scale(L.1)),
    L.2 = as.numeric(scale(L.2))
  )
K <- length(unique(df_fit$k_fac))
```

3. Bayesian Model Fitting

We now fit the joint model for the recurrent events and terminal process using Stan.

We fit a joint Bayesian model for (i) the terminal process T_k and (ii) the recurrent events Y_k, sharing the discrete time structure and history records. Each sub-model includes: the current treatment, lagged outcomes (e.g., $I(\text{lagY}_k^2)$), standardized covariates ($L.1, L.2$), and a piecewise-constant baseline (time_baseline_T[k], time_baseline_Y[k]).

Following Oganisian et al. (2024), we use β to denote coefficients in the terminal (death) model and θ to denote coefficients in the recurrent-event model. We expose variable names in diagnostics and summaries as follows:

- $\beta_{T:L.1}$, $\beta_{T:L.2}$: covariate ($L.1$, $L.2$ here) effects in the terminal (T) model.
- $\theta_{Y:L.1}$, $\theta_{Y:L.2}$: covariate ($L.1$, $L.2$ here) effects in the recurrent-event (Y) model.
- $\theta_{Y,\text{lag}:I(\text{lagYk}^2)}$: coefficients for lagged outcome features in the recurrent-event model.
- time_baseline_T[j], time_baseline_Y[j]: j-th time-interval baselines capturing secular time trends.
- treatment_effect_T, treatment_effect_Y: treatment effects for the terminal and recurrent processes, respectively.

```
# Stan fit
fit <- fit_causal_recur(
  data      = df_fit,
  K         = K,
  id_col    = "id",
  time_col  = "k_fac",
  treat_col = "Ak",
  lag_col   = "lagYk",
  formula_T = Tk ~ Ak + I(lagYk^2) + L.1 + L.2,
  formula_Y = Yk ~ Ak + I(lagYk^2) + L.1 + L.2,
```

```
  cores     = cores,
  verbose   = TRUE
)
```

```
## Loading pre-compiled Stan model from: D:/R/win-library/4.4/BayCauRETM/stan/causal_recur_model.rds
```

```
## Re-compiling Stan model from: D:/R/win-library/4.4/BayCauRETM/stan/causal_recur_model.stan
```

```
## Sampling (4 chains * 2000 iter, cores=30)...
```

4. MCMC Diagnostics

Evaluate convergence and identify any problematic chains. We report R-hat (target 1.00), effective sample size (should be large), and show traceplots grouped by model block. By default, the mcmc_diagnosis() and its plot function checks multiple model blocks, including the T-model, Y-model, treatment effects, and lag terms. However, for simplicity and clarity, users can focus on a specific component as follows. Specifically, "T-model" refers to the baseline hazard and covariate effects in the terminal event model, while "Y-model" targets the same components in the recurrent event model. "treatment_effect_T" and "treatment_effect_Y" refer to the treatment effects in the T-model and Y-model, respectively. "Lag" corresponds to the lag kernel terms that capture dependencies across time. These names can be supplied directly to the pars argument in the plot() call, and matching is case-insensitive and supports partial keyword matching for convenience. P corresponds to the columns of the design matrices L_Tk and L_Yk constructed from formula_T and formula_Y.

```
# MCMC Diagnosis
rstan::check_hmc_diagnostics(fit$stan_fit)
```

```
##
## Divergences:
```

```
## 20 of 4000 iterations ended with a divergence (0.5%).
## Try increasing 'adapt_delta' to remove the divergences.
```

```
##
## Tree depth:
```

```
## 0 of 4000 iterations saturated the maximum tree depth of 15.
```

```
##
## Energy:
```

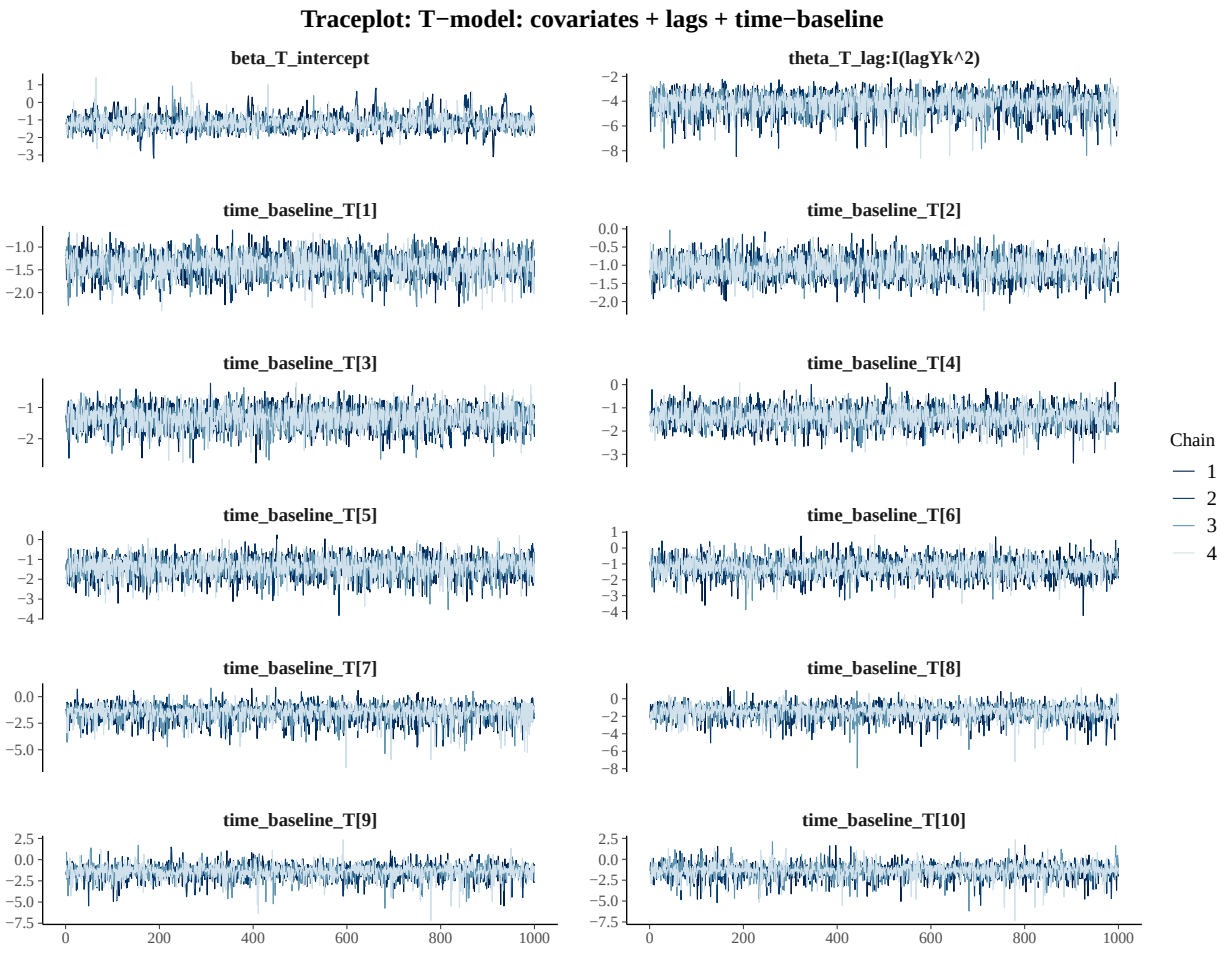
```
## E-BFMI indicated no pathological behavior.
```

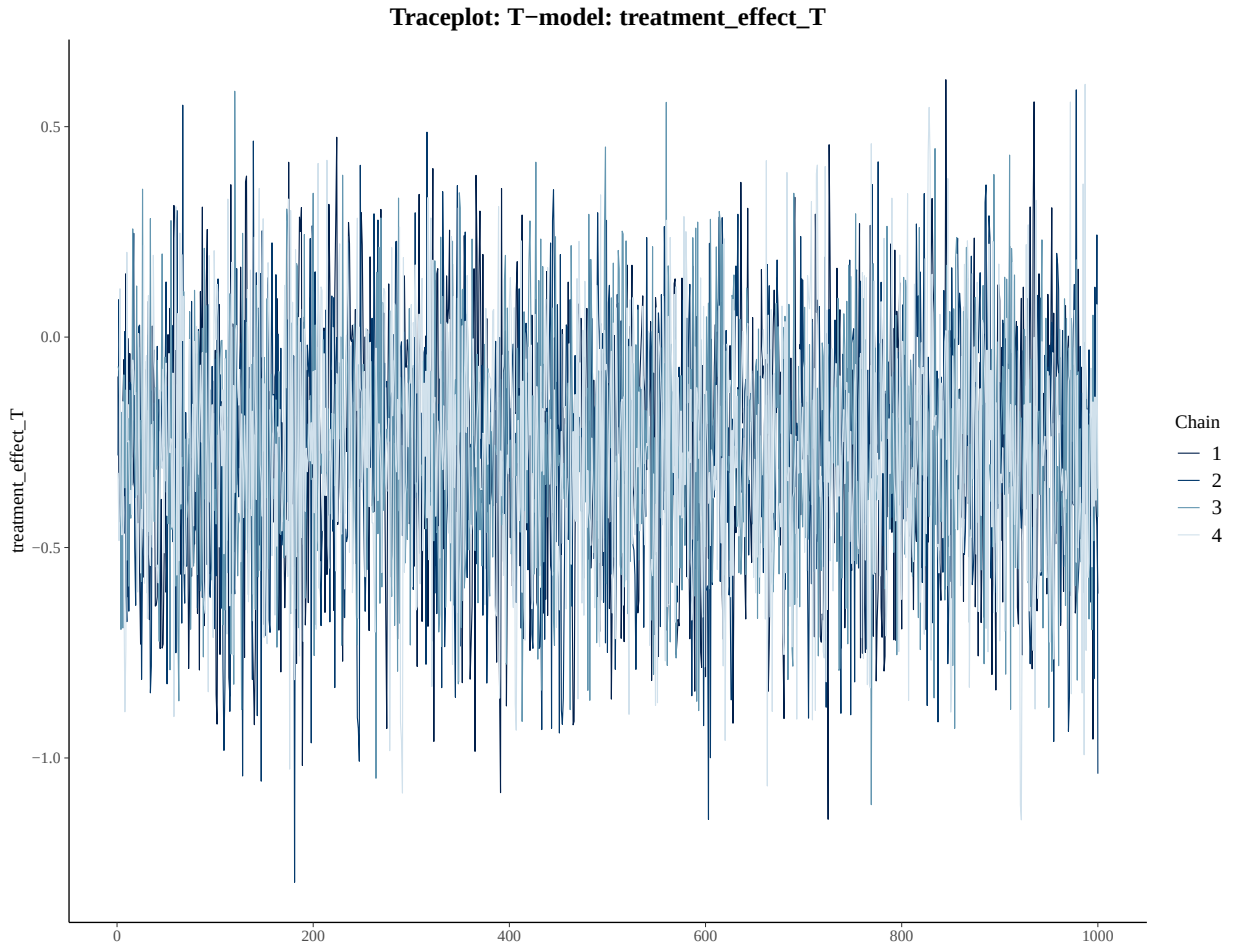
```
diag <- mcmc_diagnosis(fit)
```

```
## ----- MCMC Rhat & Effective Sample Size -----
##                 Parameter     n_eff      Rhat
## 1  theta_T_lag:I(lagYk^2) 4043.2213 0.9996728
## 2      time_baseline_T[1] 3748.7310 0.9991667
```

```
## 3      time_baseline_T[2] 4128.5189 0.9997099
## 4      time_baseline_T[3] 4875.4662 0.9997434
## 5      time_baseline_T[4] 4218.5829 0.9993660
## 6      time_baseline_T[5] 4303.6104 0.9998946
## 7      time_baseline_T[6] 4068.7230 0.9999841
## 8      time_baseline_T[7] 4138.3207 0.9997244
## 9      time_baseline_T[8] 3583.9016 1.0000311
## 10     time_baseline_T[9] 3756.8256 0.9996320
## 11    time_baseline_T[10] 3803.8491 0.9994364
## 12 theta_Y_lag:I(lagYk^2) 1471.7730 1.0063451
## 13     time_baseline_Y[1] 3990.3140 0.9997153
## 14     time_baseline_Y[2] 3601.4790 1.0038586
## 15     time_baseline_Y[3] 4888.8775 0.9993624
## 16     time_baseline_Y[4] 2953.4266 1.0037303
## 17     time_baseline_Y[5] 3802.9640 1.0012266
## 18     time_baseline_Y[6] 3178.7094 1.0019557
## 19     time_baseline_Y[7] 3655.0890 1.0011517
## 20     time_baseline_Y[8]  832.0401 1.0122300
## 21     time_baseline_Y[9] 4496.1384 1.0008715
## 22    time_baseline_Y[10] 4670.2911 1.0011264
## 23     treatment_effect_T 4099.7557 0.9995201
## 24     treatment_effect_Y 3750.7110 1.0015077
## 25     theta_lag_extra[1] 5507.9220 1.0012521
## (Values close to Rhat = 1 and large n_eff indicate good convergence.)
```

```
plot(diag, pars = "T-model")
```


Traceplot: T−model: covariates + lags + time−baseline
beta_T_intercept
theta_T_lag:I(lagYk^2)
time_baseline_T[1]
time_baseline_T[2]
time_baseline_T[3]
time_baseline_T[4]
time_baseline_T[5]
time_baseline_T[6]
time_baseline_T[7]
time_baseline_T[8]
time_baseline_T[9]
time_baseline_T[10]
0
200
400
600
800
1000
Chain
1
2
3
4




```
# Extract posterior draws for covariate coefficients (beta_L and theta_L)
post <- rstan::extract(fit$stan_fit)

# Covariate coefficient draws (iterations * P)
betaL_draws  <- post$betaL
thetaL_draws <- post$thetaL

# Posterior means and 95% Credible Interval
betaL_sum <- apply(betaL_draws, 2, function(x) c(mean=mean(x), quantile(x, c(.025,.975))))
thetaL_sum <- apply(thetaL_draws, 2, function(x) c(mean=mean(x), quantile(x, c(.025,.975))))

betaL_sum
```

```
##
##                [,1]         [,2]
##   mean  -0.1657825 -0.008283579
##   2.5%  -0.4830987 -0.324081421
##   97.5%  0.1428193  0.310061302
```

```
thetaL_sum
```

```
##
```

```
##                [,1]        [,2]
##   mean  -0.01901186 -0.02913459
##   2.5%  -0.08999540 -0.09913027
##   97.5%  0.05097030  0.04054337
```

5. G-Computation

We evaluate treatment–start strategies $s \in \mathcal{S}$ over intervals $k = 1, \ldots, K$ using Bayesian g-computation with death as a terminal event. For subject i, under strategy s we simulate recurrent counts $Y_{ik}^{(m)}(s)$ and the not-at-risk indicator $T_{ik}^{(m)}(s) \in \{0, 1\}$ (equals 1 from the first interval after death). Let L_i be the subject's baseline covariate vector. Our estimand is

$$\theta_s(K) = \mathbb{E}\left[\frac{\sum_{k=1}^{K} Y_{ik}^{(m)}(s)}{K - \sum_{k=1}^{K} T_{ik}^{(m)}(s)}\right],$$

and the contrast $\Delta(s, s') = \theta_s(K) - \theta_{s'}(K)$. In the package we set $s' = K + 1$ as the reference ("never treat").

For each posterior draw m, g_computation() approximates the conditional expectation given L_i by averaging over B Monte-Carlo replicates:

$$R_i^{(m)}(s) = \frac{1}{B}\sum_{b=1}^{B}\left[\frac{\sum_{k=1}^{K} Y_{ikb}^{(m)}(s)}{K - \sum_{k=1}^{K} T_{ikb}^{(m)}(s)}\right].$$

Then it integrates over the baseline distribution of L using Dirichlet weights $\pi_i^{(m)}$ (Bayesian bootstrap):

$$\theta_s^{(m)}(K) = \sum_{i=1}^{n} \pi_i^{(m)} R_i^{(m)}(s), \qquad \Delta^{(m)}(s) = \theta_s^{(m)}(K) - \theta_{K+1}^{(m)}(K).$$

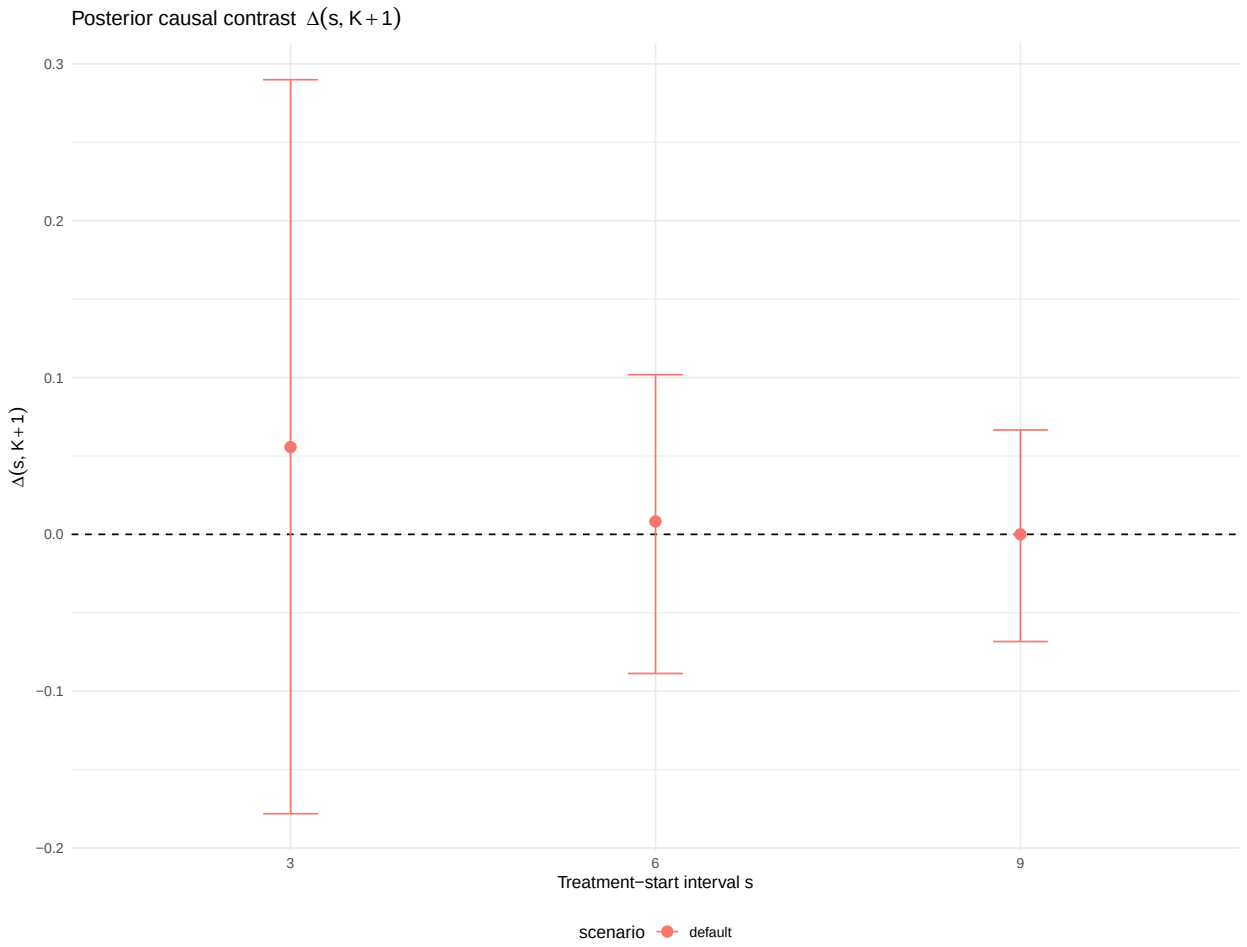
Posterior draws $\{\Delta^{(m)}(s)\}_m$ are summarized by their mean and 95% credible interval. For clarity, although we construct a baseline covariate data frame below for illustration, the g_computation() function performs marginalization over baseline covariates internally using Bayesian bootstrap weights, and this object is not explicitly passed to the function.

```
baseline_df <- fit$data_preprocessed %>%
  group_by(pat_id) %>%
  slice_min(order_by = k_idx, n = 1) %>%
  arrange(pat_id) %>%
  ungroup()

# Note: baseline_df is created for illustration only.
# The g_computation() function marginalizes over baseline covariates internally
# via Bayesian bootstrap weights and does not require this object as input.

gcomp <- g_computation(
  fit_out = fit,
  s_vec   = s_vec,
  B       = B,
  cores   = 1
)
```

```r
print(gcomp)
```

```
## Causal contrast delta(s, K+1) summary:
##  s         Mean       X2.5.     X97.5.
##  3 5.568411e-02 -0.17814240 0.28995333
##  6 8.265013e-03 -0.08869771 0.10186127
##  9 2.044594e-05 -0.06834766 0.06657815
```

```r
plot(gcomp, ref_line = 0)
```




```r
plot(gcomp, interactive = TRUE, ref_line = 0)
```

6. Switching Probability Summary

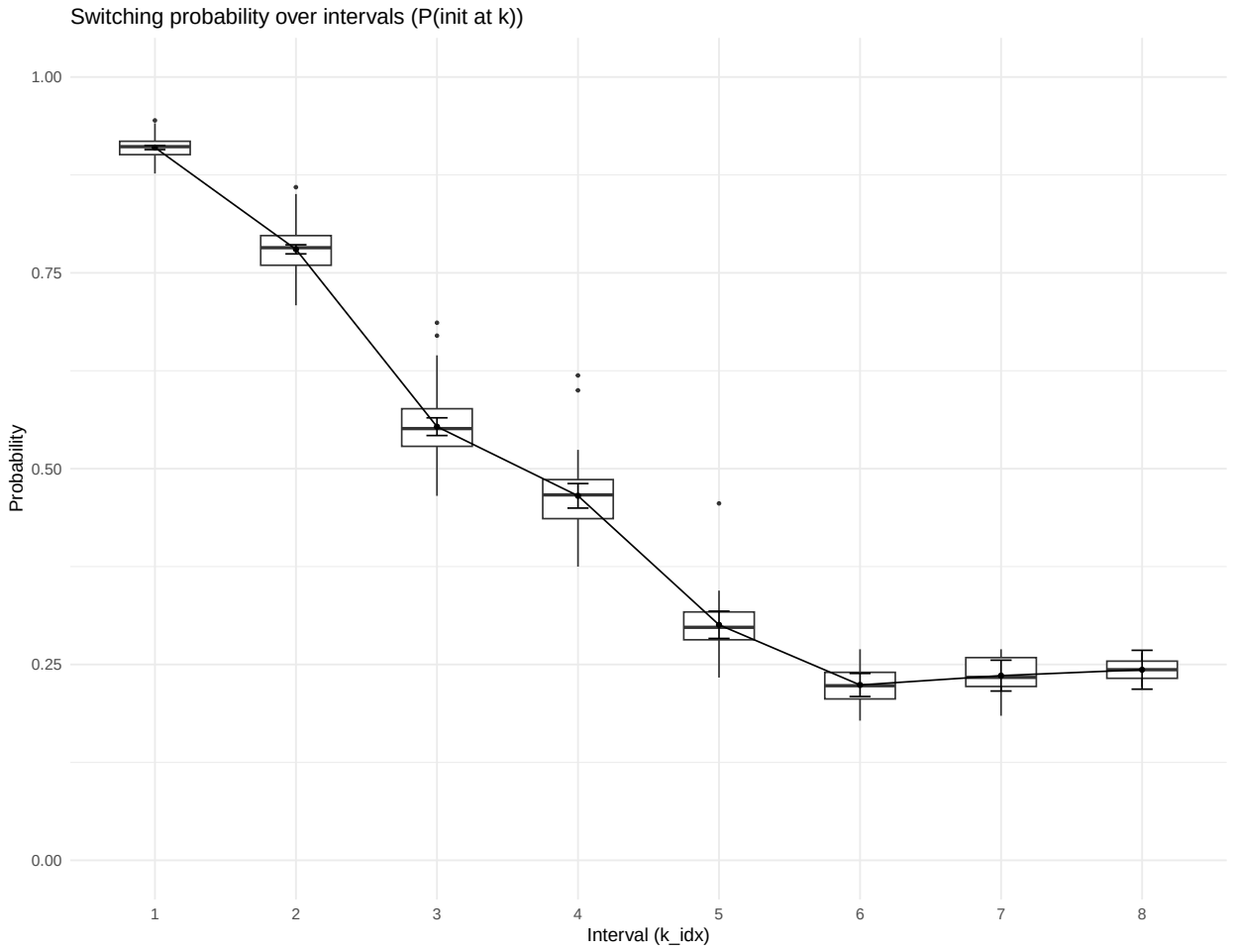
The switching probability summary quantifies how often subjects switch treatment across follow-up intervals. For each interval k, we estimate the within-subject probability of switching.

The switching probability summary quantifies how often subjects change treatment across follow-up intervals. There are two possible scales:

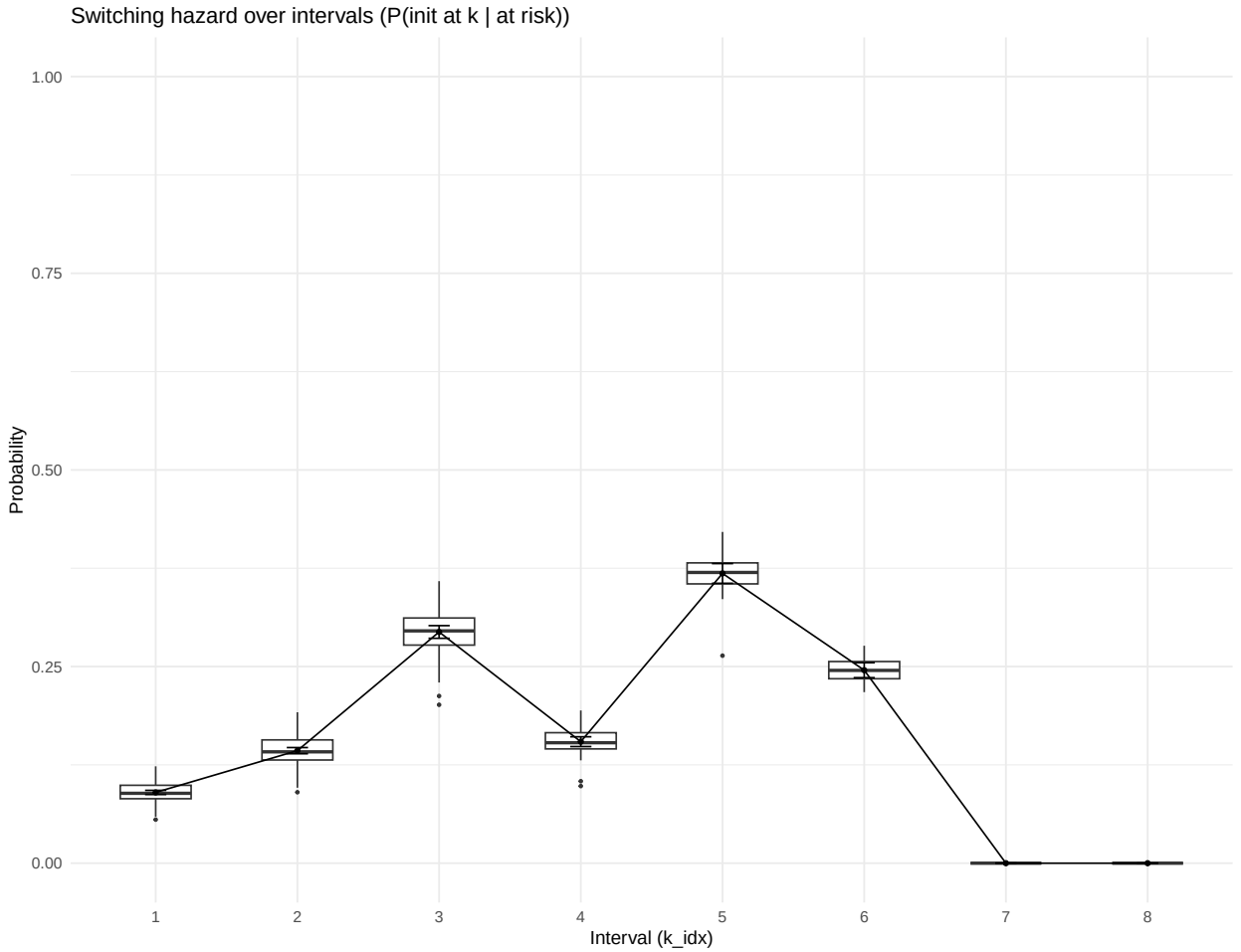
- Mass (default): the marginal probability that switching occurs at interval k, which shows how many subjects in the population switched at each time point.
- Hazard: the conditional probability of switching at interval k given that the subject has not yet switched before k.

By default, the function uses scale="mass":

```
# default (scale="mass")
sw  <- switching_probability_summary(fit$data_preprocessed, covariates = c("L.1", "L.2"), scale="mass")
plot(sw)
```




```
sw  <- switching_probability_summary(fit$data_preprocessed, covariates = c("L.1", "L.2"), scale="hazard
plot(sw)
```



7. Summary of Causal Estimates

This table summarizes posterior distributions of model parameters and g-computation estimates. Parameter tables report Mean, 2.5%, 97.5%, R-hat, n_eff, MCSE, and CI width. Effects close to 0 with tight intervals imply little evidence of impact. Baseline blocks (time_baseline_*) quantify secular time risk; do not interpret them as treatment effects.

```
sum_tbl <- result_summary_table(
  fit_out   = fit,
  gcomp_out = gcomp,
  s_vec     = s_vec,
  format    = "kable"
)
print(sum_tbl)
```

```
## ----- Posterior Parameters -----
##
##
## |Parameter              |      Mean|     X2.5.|    X97.5.|     Rhat|     n_eff|     MCSE| CI_width|
## |:----------------------|---------:|---------:|---------:|--------:|---------:|--------
:|--------:|
```

```
## |theta_T_lag:I(lagYk^2) | -4.163495| -6.156514| -2.730886| 0.999673| 4043.2213| 0.013650| 3.425628|
## |time_baseline_T[1]     | -1.403884| -1.945598| -0.912718| 0.999167| 3748.7310| 0.004248| 1.032880|
## |time_baseline_T[2]     | -1.085123| -1.650292| -0.492085| 0.999710| 4128.5189| 0.004615| 1.158207|
## |time_baseline_T[3]     | -1.340576| -2.059847| -0.696947| 0.999743| 4875.4662| 0.005001| 1.362900|
## |time_baseline_T[4]     | -1.371349| -2.224765| -0.576268| 0.999366| 4218.5829| 0.006356| 1.648497|
## |time_baseline_T[5]     | -1.393497| -2.469617| -0.515554| 0.999895| 4303.6104| 0.007363| 1.954062|
## |time_baseline_T[6]     | -1.228294| -2.389829| -0.182637| 0.999984| 4068.7230| 0.008422| 2.207192|
## |time_baseline_T[7]     | -1.514010| -3.389412| -0.336074| 0.999724| 4138.3207| 0.011709| 3.053339|
## |time_baseline_T[8]     | -1.468155| -3.288208| -0.159727| 1.000031| 3583.9016| 0.013100| 3.128481|
## |time_baseline_T[9]     | -1.414690| -3.302739|  0.010557| 0.999632| 3756.8256| 0.012975| 3.313296|
## |time_baseline_T[10]    | -1.379758| -3.266730|  0.114876| 0.999436| 3803.8491| 0.013388| 3.381606|
## |treatment_effect_T     | -0.268343| -0.832832|  0.283079| 0.999520| 4099.7557| 0.004439| 1.115911|
## |theta_Y_lag:I(lagYk^2) | -2.212505| -2.846122| -1.642556| 1.006345| 1471.7730| 0.008031| 1.203566|
## |time_baseline_Y[1]     |  1.097261|  0.981489|  1.209320| 0.999715| 3990.3140| 0.000919| 0.227831|
## |time_baseline_Y[2]     |  1.038567|  0.852173|  1.211018| 1.003859| 3601.4790| 0.001519| 0.358845|
## |time_baseline_Y[3]     |  1.047825|  0.844407|  1.242295| 0.999362| 4888.8775| 0.001468| 0.397888|
## |time_baseline_Y[4]     |  0.958851|  0.693154|  1.202924| 1.003730| 2953.4266| 0.002398| 0.509769|
## |time_baseline_Y[5]     |  1.071126|  0.769642|  1.362145| 1.001227| 3802.9640| 0.002475| 0.592503|
## |time_baseline_Y[6]     |  1.243500|  0.901395|  1.612853| 1.001956| 3178.7094| 0.003291| 0.711458|
## |time_baseline_Y[7]     |  1.272860|  0.811835|  1.760846| 1.001152| 3655.0890| 0.004064| 0.949011|
## |time_baseline_Y[8]     |  1.086027|  0.428340|  1.649305| 1.012230|  832.0401| 0.010501| 1.220965|
## |time_baseline_Y[9]     |  1.238499|  0.575628|  1.983811| 1.000872| 4496.1384| 0.005271| 1.408183|
## |time_baseline_Y[10]    |  0.904039| -0.147061|  1.666172| 1.001126| 4670.2911| 0.006516| 1.813233|
## |treatment_effect_Y     | -0.007025| -0.159476|  0.142381| 1.001508| 3750.7110| 0.001251| 0.301857|
##
## ----- g-computation delta(s, K+1) -----
##
##
## |  s|     Mean|     X2.5.|   X97.5.| CI_width|
## |--:|--------:|---------:|--------:|--------:|
## |  3| 0.055684| -0.178142| 0.289953| 0.468096|
## |  6| 0.008265| -0.088698| 0.101861| 0.190559|
## |  9| 0.000020| -0.068348| 0.066578| 0.134926|
```