The Journal of Open Source Software

# BayCauRETM: R package for Bayesian Causal Inference for Recurrent Event Outcomes

**Yuqin Wang** [1], **Keming Zhang** [1], **and Arman Oganisian** [1¶]

**1** Department of Biostatistics, Brown University, Providence, RI, United States ¶ Corresponding author

**DOI:** 10.xxxxxx/draft

**Software**

- Review
- Repository
- Archive

**Editor:** Adithi R Upadhya

**Reviewers:**

- @jackmwolf
- @larryshamalama

**Submitted:** 26 September 2025
**Published:** unpublished



## Summary

Observational studies are often conducted to estimate the causal effect of medical treatments on the average rate of a recurrent event outcome within a specific follow-up window in a defined target population. Recurrent events ( e.g. hospitalizations, relapses, and infections) may occur multiple times during follow-up. Causal estimation is challenging with such outcomes because: 1) Recurrent events are typically jointly observed with a terminal event (e.g., death), which precludes future recurrences. 2) Both event count process and the terminal process are unobserved after possible dropout - leading to right censored total event counts and survival time. 3) Patients may initiate the treatment of interest at different times, yielding as many strategies as possible initiation times. Finally, 4) patients are not assigned to treatment strategies randomly, so formal causal methods are required to adjust for observed confounders - i.e. drivers of both treatment patterns and either the recurrent event or death processes.

This paper presents `BayCauRETM`, an `R` package for Bayesian estimation of causal effects of different treatment initiation strategies on a recurrent event outcome in the presence of death and censoring. It does so by implementing the methodology developed by Oganisian et al. (2024) which uses Bayesian statistical methods within the potential outcomes causal inference framework. Users specify an initiation time and supply a `data.frame` containing confounders and columns for censoring, death, and interval-specific recurrent counts, then define terminal and recurrent models via standard `R` formula syntax. Given these inputs, `BayCauRETM` performs causal adjustment and outputs adjusted expected event rates over follow-up under the specified initiation time. The package also provides diagnostic and visualization utilities.

Intended users include statisticians, epidemiologists, and health-services researchers analyzing observational data.

## Statement of need

The usual methods (Andersen & Keiding, 2002; Cowling et al., 2006; Ghosh & Lin, 2002) for time-to-event outcomes, recurrent-event outcomes, and multi-state outcomes remain useful for descriptive or associational analysis of death and event rates. Many of these methods can be implemented in `R` via base functions such as `glm()`, GEE functions in the `geepack` package, or survival functions in the `survival` package. Some specialized `R` packages such as `msm` (Jackson, 2011) and `reReg` (Chiou et al., 2023) can implement these methods directly. However, while descriptively useful, these methods generally do not target causal effects.

Recent methods targeting causal effects have used inverse-weighted approaches (Schaubel & Zhang, 2010), outcome modeling approaches (Janvin et al., 2024), and doubly-robust approaches (Su et al., 2020). In terms of software, no dedicated `R` packages are available for these papers beyond implementation code made available as accompanying supplementary information. Moreover, these do not accommodate complexities (1)-(4) listed in the previous

section simultaneously.

Oganisian et al. (2024) developed statistical methods that handle complexities (1)-(4) and conducted a thorough simulation-based validation of these methods. However, the focus was on methodological development and validation, and so development of a user-friendly, off-the-shelf `R` package with readable help files was out of scope. `BayCauRETM` fills this gap by operationalizing the methods of Oganisian et al. (2024). Its syntax seeks to maximize familiarity to base `R` users by mirroring standard regression functions such as `lm()` and `glm()`. It also has extensive help pages accessible via `help()` or the `?` command. Thus, `BayCauRETM` provides the first user-friendly software for analyzing complex recurrent-event data while handling complexities (1)-(4) described in the Summary section above.

## Data structure, model, and outputs

In this section, we provide an overview of the expected input data structure, models that are run under-the-hood, and expected outputs. We refer readers to Oganisian et al. (2024) for methodological details.

### Data structure and preprocessing

The package expects longitudinal data in long, person-interval format. For follow-up time $\tau$, the window $[0, \tau)$ is partitioned into $K$ equal-length intervals $I_k = [\tau_{k-1}, \tau_k)$ for $k = 1, \dots, K$ with $\tau_0 = 0$ and $\tau_K = \tau$. Each row represents a patient-interval; a subject has one row per interval at risk.

Required `data.frame` variables are: subject ID, interval index $k$, treatment indicator (0 until the initiation interval, then 1), interval-specific count of recurrent events, and a terminal-event indicator (0 up to death, 1 thereafter). Optional variables include baseline covariates and lagged history (e.g., one-interval lag of the event count).

Each row contains a monotone death indicator at the start of interval $k$, $T_k$, a monotone treatment indicator by the end of interval $k$, $A_k$, the interval count $Y_k$ and baseline covariates $L \in \mathcal{L}$.

### Causal estimand and potential outcomes

Let $a(s) = (\underbrace{0, \dots, 0}_{s-1}, 1, \dots, 1)$ be the strategy that initiates treatment at interval $s \in \{1, 2, \dots, K+1\}$. Let $T_k^{a(s)}$ and $Y_k^{a(s)}$ denote, respectively, the death indicator and number of recurrent events that would have been observed in interval $k$ under strategy $a(s)$. Formally, $T_k^{a(s)}$ is the potential death indicator following sequence $a(s)$ up to interval $k$, and $Y_k^{a(s)}$ is the corresponding potential number of events.

The target is the difference in average potential incidence rates over follow-up under two initiation times:

$$\Delta(s, s') = \mathbb{E}\left[\frac{\sum_{k=1}^{K} Y_k^{a(s)}}{K - \sum_{k=1}^{K} T_k^{a(s)}}\right] - \mathbb{E}\left[\frac{\sum_{k=1}^{K} Y_k^{a(s')}}{K - \sum_{k=1}^{K} T_k^{a(s')}}\right].$$

### Model specification

The package runs a pair of discrete-time models conditional on shared treatment and covariate terms.

Here and throughout, we use overbar notation to denote the full history of the recurrent event process up to the previous interval, i.e., $\bar{Y}_{k-1} = (Y_1, Y_2, \dots, Y_{k-1})$.

1. Discrete-time hazard model for the terminal event that models death at a given interval conditional on survival up to that interval:

$$\lambda_k(a_k, \bar{Y}_{k-1}, l) = \Pr(T_k = 1 \mid T_{k-1} = 0,\, a_k, \bar{Y}_{k-1}, l).$$

2. Distribution for the number of event occurrences in a given interval conditional on survival through that interval:

$$f(y_k \mid a_k, \bar{Y}_{k-1}, l) = \Pr(Y_k = y_k \mid T_k = 0,\, a_k, \bar{Y}_{k-1}, l).$$

Here, $f(y_k \mid a_k, \bar{Y}_{k-1}, l)$ denotes the Poisson probability mass function with conditional mean (intensity) $\mu_k(a_k, \bar{Y}_{k-1}, l) = \mathbb{E}[Y_k \mid A_k, \bar{Y}_{k-1}, L]$. Together, these two models multiply to form a joint model for the terminal and recurrent event occurrence at a given interval.

The functions in `BayCauRETM` implement the following models for the hazard and intensity, respectively:

$$\begin{aligned}\text{logit}\, \lambda_k(a_k, \bar{Y}_{k-1}, l) &= \beta_{0k} + l^\top \beta_L \\ &\quad + y_{k-1}\beta_Y + \beta_A a_k, \\ \log \mu_k(a_k, \bar{Y}_{k-1}, l) &= \theta_{0k} + l^\top \theta_L + y_{k-1}\theta_Y + \theta_A a_k.\end{aligned}$$

The time-varying intercepts $\{\beta_{0k}\}$ and $\{\theta_{0k}\}$ parameterize the baseline hazard and event intensity, respectively. They are assigned a first-order autoregressive (AR1) smoothing prior. See Oganisian et al. (2024) for more details.

### Posterior inference and g-computation

`BayCauRETM` conducts full posterior inference for the joint models using Stan (Carpenter et al., 2017) through the `rstan` interface, since the posterior distribution is not available in closed form. Stan is a probabilistic programming language that implements Hamiltonian Monte Carlo to generate posterior draws.

For each parameter draw obtained from Stan, `BayCauRETM` simulates the joint death-recurrent process under $a(s)$ and $a(s')$ to obtain a posterior draw of $\Delta(s, s')$. Reporting over many draws yields posterior samples of $\Delta(s, s')$, as described by Oganisian et al. (2024). The posterior mean and the 95% credible interval (2.5th and 97.5th percentiles) are reported.

## Quickstart

Below we provide a minimal, copy-pastable example illustrating the core functionality of `BayCauRETM`: fitting a joint recurrent-event and terminal-event model and estimating causal effects under alternative treatment initiation strategies. A small example dataset is shipped with the package, allowing users to run the example without any external data dependencies.

### Installation and setup

We first install the development version of the package from GitHub and load the required libraries. For reproducibility, we also set a random seed.

```
# install.packages("pak")
pak::pak("LnnnnYW/BayCauRETM")

library(BayCauRETM)
library(dplyr)
library(tidyr)

set.seed(123)
```

### Data loading

The package ships with a small example dataset used in the demonstration code. The dataset is stored in the package directory and can be loaded using `system.file()`. Loading this file creates a data frame named `df` in the workspace.

```
# Load the example dataset shipped with the package
rdata_path <- system.file("demo_code", "data.Rdata", package = "BayCauRETM")
stopifnot(file.exists(rdata_path))
load(rdata_path)  # loads an object named `df`
```

### Minimal preprocessing

The data are assumed to be in long, person-interval format. For a fast illustrative run, we subset the data to a small number of subjects and perform minimal preprocessing. This includes ordering observations by subject and time, constructing a discrete time index, creating a lagged event-count variable, removing incomplete cases, and standardizing continuous covariates. These steps mirror the preprocessing required for real applications but are kept intentionally simple here.

```
# Minimal preprocessing
df_fit <- df %>%
  filter(id %in% 1:50) %>%                    # subset for quickstart
  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 =
  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))
```

### Model fitting

We next fit the joint Bayesian model for the recurrent-event and terminal-event processes using the main function `fit_causal_recur()`. The user specifies the outcome models through standard R formula syntax, along with the relevant column names for subject ID, time index, treatment, and lagged history. Here we use a single core and suppress verbose output for speed.

```
# Fit the joint recurrent + terminal event model (small settings for illustration)
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     = 1,
  verbose   = FALSE
)
```

**Causal effect estimation via g-computation**

Finally, we estimate causal effects corresponding to different treatment initiation strategies using Bayesian g-computation. In this example, we compare initiation at two different time points. The output summarizes posterior draws of the causal estimand, including point estimates and uncertainty intervals.

```r
# Bayesian g-computation for two treatment-start strategies
gcomp <- g_computation(
  fit_out = fit,
  s_vec   = c(3, 6),   # start at time 3 vs 6
  B       = 20,
  cores   = 1
)

print(gcomp)
```

This quickstart demonstrates the full workflow of BayCauRETM, from data preparation and model fitting to causal effect estimation. Detailed usage and example results are available on GitHub (see the demo PDF).

## Acknowledgements

This work was partially funded by the Patient Centered Outcomes Research Institute (PCORI) Contract ME-2023C1-31348.

## References

Andersen, P. K., & Keiding, N. (2002). Multi-state models for event history analysis. *Statistical Methods in Medical Research*, *11*(2), 91–115. https://doi.org/https://doi.org/10.1191/0962280202SM276ra

Carpenter, B., Gelman, A., Hoffman, M. D., Lee, D., Goodrich, B., Betancourt, M., Brubaker, M., Guo, J., Li, P., & Riddell, A. (2017). Stan: A probabilistic programming language. *Journal of Statistical Software*, *76*(1), 1–32. https://doi.org/10.18637/jss.v076.i01

Chiou, S. H., Xu, G., Yan, J., & Huang, C.-Y. (2023). Regression modeling for recurrent events possibly with an informative terminal event using r package reReg. *Journal of Statistical Software*, *105*(5), 1–34. https://doi.org/10.18637/jss.v105.i05

Cowling, B. J., Hutton, J. L., & Shaw, J. E. H. (2006). Joint modelling of event counts and survival times. *Journal of the Royal Statistical Society: Series C (Applied Statistics)*, *55*(1), 31–39. https://doi.org/https://doi.org/10.1111/j.1467-9876.2005.00529.x

Ghosh, D., & Lin, D. Y. (2002). Marginal regression models for recurrent and terminal events. *Statistica Sinica*, 663–688.

Jackson, C. (2011). Multi-state models for panel data: The msm package for r. *Journal of Statistical Software*, *38*, 1–28. https://doi.org/https://doi.org/10.18637/jss.v038.i08

Janvin, M., Young, J. G., Ryalen, P. C., & Stensrud, M. J. (2024). Causal inference with recurrent and competing events. *Lifetime Data Analysis*, *30*(1), 59–118. https://doi.org/10.1007/s10985-023-09594-8

Oganisian, A., Girard, A., Steingrimsson, J. A., & Moyo, P. (2024). A Bayesian framework for causal analysis of recurrent events with timing misalignment. *Biometrics*, *80*(4), ujae145. https://doi.org/10.1093/biomtc/ujae145

Schaubel, D. E., & Zhang, M. (2010). Estimating treatment effects on the marginal recurrent

event mean in the presence of a terminating event. *Lifetime Data Analysis*, *16*(4), 451–477. https://doi.org/10.1007/s10985-009-9149-x

Su, C.-L., Steele, R., & Shrier, I. (2020). Doubly robust estimation and causal inference for recurrent event data. *Statistics in Medicine*, *39*(17), 2324–2338. https://doi.org/https://doi.org/10.1002/sim.8541