JM2survClayton: User Guide

Jiming Chen

2026-02-26

Contents

1 Description

`JM2survClayton` fits a **generalized linear mixed-effects joint model** that links:

- a **longitudinal outcome** (Gaussian LMM or Poisson GLMM),
- **two correlated time-to-event endpoints** (Cox-type proportional hazards submodels),
- and the **dependence between the two event times** via a **Clayton survival copula**,

through **shared subject-specific random effects**. Estimation is performed using a **Monte Carlo EM (MCEM)** algorithm, with C++ backends for computational efficiency, and the implementation can accommodate **multi-dimensional random effects**.

The modeling framework and estimation strategy follow the joint modeling literature for longitudinal and bivariate survival outcomes; for methodological details, see *Generalized linear mixed-effects joint model for longitudinal and bivariate survival data*.

2 Installation and loading

```
devtools::install_github("Jimmy-Chen1249/JM2survClayton")
library(JM2survClayton)
```

3 Packaged example datasets

The package ships **four toy simulation datasets** to illustrate typical workflows under different longitudinal distributions and random-effects dimensions.

Datasets (object names):

- `Gaussian_Clayton_One`: Gaussian longitudinal outcome; random effects dimension = 1 (random intercept).
- `Poisson_Clayton_One`: Poisson longitudinal outcome; random effects dimension = 1 (random intercept).
- `Gaussian_Clayton_Two`: Gaussian longitudinal outcome; random effects dimension = 2 (random intercept + random slope).
- `Poisson_Clayton_Two`: Poisson longitudinal outcome; random effects dimension = 2 (random intercept + random slope).

Common simulation settings

- Sample size: `n = 200`.
- Censoring level: approximately `CR = 0.3`.
- The true parameter values follow the setup in *Generalized linear mixed-effects joint model for longitudinal and bivariate survival data* (see the paper for the full list).

3.1 Loading a dataset

Each dataset loads as an object with the **same name**. You can inspect its structure as follows:

```
data("Poisson_Clayton_Two")
str(Poisson_Clayton_Two, max.level = 2)
```

4 Main function `mjoint_Clayton()`

`mjoint_Clayton()` is the main user-facing function in **JM2survClayton**. It fits a joint model linking

- one longitudinal process (Gaussian LMM or Poisson GLMM),
- two Cox-type survival submodels (two event times),
- dependence between the two event times via a Clayton survival copula,

through shared subject-specific random effects, using a Monte Carlo EM (MCEM) algorithm.

Function signature:

```
mjoint_Clayton(Mixeffect, Longdat, Surdat1, Surdat2,
               formSurv1, formSurv2,
               timevars1, timevars2,
               y_type = c("gaussian","poisson"),
               rho0 = 0.5,
               nMC = 100,
               iter_max = 500,
               tol = 5e-3,
               rho.region = c(1e-4, 30),
               eps_prob = 1e-100,
               verbose = FALSE)
```

4.1 Key inputs

- Mixeffect:

  Mixed-effects formula for the longitudinal outcome (random intercept, random slope, or higher-dimensional random effects).

- Longdat:

  Longitudinal data.frame containing at least: id, time, y, and the covariates used in Mixeffect.

- Surdat1, Surdat2:

  Survival data.frames for endpoint 1 and endpoint 2. Each should contain at least: id, surtime, cens, baseline covariates used in formSurv*, and any variables referenced inside your time-varying functions (commonly B1,B2,B3 in the packaged examples).

- formSurv1, formSurv2:

  Cox-type formulas, e.g. Surv(surtime, cens) ~ W.

- y_type:

  "gaussian" or "poisson" for the longitudinal submodel.

- rho0:

  Initial value for the Clayton copula parameter rho (> 0).

- nMC, iter_max, tol, rho.region, eps_prob, verbose:

  MCEM/optimization controls.

4.2 Time-varying covariates interface

For each endpoint, provide `timevars1` and `timevars2` as a named list:

timevars <- list(fixed = FUN, rand = FUN)

- fixed(t, index, Sur = NULL, ...):

  Returns the time-varying covariates for the *fixed-effects* part of the survival model. Output must be a numeric matrix with nrow = length(t) and ncol = p_k (p_k >= 1).

- rand(t, index, ...):

  Returns the time-varying *random-effect design* used in the association term. Output must be a numeric matrix with nrow = length(t) and ncol = r, where r matches the random-effect dimension implied by `Mixeffect`.

Minimal template:

timevars1 <- list(fixed = function(t, index, Sur = NULL, ...) { as.matrix(...) }, rand = function(t, index, ...) { as.matrix(...) })

(Same for `timevars2`.)

4.3 Returned value

An object of class `mjoint_fit`, typically containing:

- `theta`: estimated parameters (longitudinal + survival + copula rho),
- `iter`: number of EM iterations used,
- plus additional components depending on your implementation.

Use:

- `summary(fit)` (S3 method `summary.mjoint_fit()`)

5 Fitting a model (example)

Below is a complete, concrete example using the packaged dataset `Poisson_Clayton_Two` (Poisson longitudinal + 2D random effects).

```r
library(JM2survClayton)

## 1) Load packaged dataset
data("Poisson_Clayton_Two")  # loads an object named `Poisson_Clayton_Two`

Longdat <- Poisson_Clayton_Two$Longdat
Surdat1 <- Poisson_Clayton_Two$Surdat1
Surdat2 <- Poisson_Clayton_Two$Surdat2

## 2) Specify model formulas
Mixeffect <- y ~ time + X1 + X2 + (1 + time | id)   # 2D RE: intercept + slope
formSurv1 <- survival::Surv(surtime, cens) ~ W
formSurv2 <- survival::Surv(surtime, cens) ~ W

## 3) Define time-varying covariates for each endpoint
## fixed: piecewise-constant using (B1,B2,B3) stored in Surdatk
## rand : (1, t) to match the 2D random effects in Mixeffect
make_timevars <- function(re_dim = 2) {
  fixed_fun <- function(t, index, Sur = NULL, ...) {
    B1 <- Sur$B1[index]; B2 <- Sur$B2[index]; B3 <- Sur$B3[index]
    z  <- B1 * (t <= B3) + B2 * (t > B3)
    matrix(z, ncol = 1)
  }
  rand_fun <- function(t, index, ...) {
    if (re_dim == 1) matrix(1, nrow = length(t), ncol = 1) else cbind(1, t)
  }
  list(fixed = fixed_fun, rand = rand_fun)
}

timevars1 <- make_timevars(re_dim = 2)
timevars2 <- make_timevars(re_dim = 2)

## 4) Fit the joint model
set.seed(1)

fit <- mjoint_Clayton(
  Mixeffect  = Mixeffect,
  Longdat    = Longdat,
  Surdat1    = Surdat1,
  Surdat2    = Surdat2,
  formSurv1  = formSurv1,
  formSurv2  = formSurv2,
  timevars1  = timevars1,
  timevars2  = timevars2,
  y_type     = "poisson",
  rho0       = 0.5,
  nMC        = 100,
  iter_max   = 200,
  tol        = 5e-3,
```

```
  rho.region = c(1e-4, 30),
  verbose    = TRUE
)

summary(fit)
```

6 Other functions

- Internal utilities: `build_surv_block()`, `initsSurv()`, `run_em()`, `stepEM_Clayton_Gaussian()`, `stepEM_Clayton_Poisson()`, and C++ backends.

7 Notes

- **Reproducibility:** set a random seed before fitting, e.g. `set.seed(1)`, since MCEM uses Monte Carlo sampling.
- **IMPORTANT:** `timevars1` and `timevars2` are **user-defined**. You must set them to match **your own case/design**:
  - If your time-varying covariates are not based on `(B1, B2, B3)`, update `fixed_fun()` accordingly.
  - If the random-effects structure in `Mixeffect` changes (e.g. `(1|id)` vs `(1+time|id)` or higher dimension), you must update `rand_fun()` so that its number of columns matches the random-effect dimension.
- A quick sanity check: for each endpoint, `fixed_fun(t, i, Sur=...)` should return a `length(t) × p_k` matrix, and `rand_fun(t, i)` should return a `length(t) × r` matrix, where `r` matches the random-effect dimension in `Mixeffect`.