discfrail

Francesca Gasperoni, Christopher Jackson

2025-06-17

Contents

Introduction **1**

Notation and Model **1**

Simulation of hierarchical time-to-event data with a latent clustering structure of groups **3**

Example: Weibull baseline hazard . . . 4

Example: Generic baseline hazard . . . 7

Cox model with a nonparametric discrete frailty **8**

References **15**

Introduction

Package **discfrail** (**disc**rete **frail**ty) is an R package that provides a novel, flexible model for hierarchical time-to-event data in which a clustering structure of groups is suspected. The random effects that define the group-level frailties follow a nonparametric discrete distribution, and the baseline hazards are left unspecified, as in the Cox model. The package contains a function to fit the Cox model with nonparametric discrete frailty term and two functions for simulating data. Full details of the methods are given in Gasperoni *et al.* (2018). This vignette gives a brief overview of the methods, and worked examples of using the functions.

Notation and Model

Consider a random sample with a hierarchical structure, i.e, where each individual subject, or statistical unit, belongs to one group. Define T^*_{ij} as the survival time and C_{ij} as the censoring time of subject i, $i = 1, ..., n_j$, in the j-th group, $j = 1, ..., J$. Let $\mathbf{X}_{ij} = (X_{ij1}, ..., X_{ijp})^T$ be a vector of covariates, assumed constant over time, for subject i in group j. Then, we define $T_{ij} = min(T^*_{ij}, C_{ij})$, t_{ij} its realization and $\delta_{ij} = \mathbf{1}_{(T^*_{ij} \leq C_{ij})}$.

Let $\tilde{\mathbf{w}}$ be a vector of shared random effects, and $\mathbf{w} = \exp\{\tilde{\mathbf{w}}\}$, be the corresponding vector of shared frailties that represent unexplained relative hazards between the groups. Conventionally, frailties are assumed to follow a parametric distribution, typically the Gamma. However in `discfrail`, the frailties are assumed to have a nonparametric, discrete distribution, with an unknown number of points in the support. This allows an arbitrarily flexible random effects distribution, and gives a method for clustering. In particular, we assume that each group j can belong to one latent population k, $k = 1, ..., K$, with probability π_k. In this case, $w_1, ..., w_K$ are the corresponding distinct values for the frailties in each population, where $\mathbb{P}\{w = w_k\} = \pi_k$.

We introduce also an auxiliary (latent) indicator random variable z_{jk} which is equal to 1 if the j-th group belongs to the k-th population, so, $z_{jk} \overset{i.i.d}{\sim} Bern(\pi_k)$. Since each group belongs to only one population, $\sum_{k=1}^{K} z_{jk} = 1$ for each j.

Therefore, in the nonparametric discrete frailty Cox model of `discfrail`, the hazard for individual i in group j, conditional on $\mathbf{w}$ and on the z_{jk} is:

$$\lambda(t; \mathbf{X}_{ij}, w_k, z_{jk}) = \prod_{k=1}^{K} \left[\lambda_0(t) w_k \exp(\mathbf{X}_{ij}^T \boldsymbol{\beta})\right]^{z_{jk}}, \tag{1}$$

where $\lambda_0(t)$ represents the baseline hazard, $\boldsymbol{\beta}$ is the vector of regression coefficients and w_k is the frailty term shared among groups of the same latent population k. Since the baseline hazard is unspecified, and the remaining parameters are estimated by maximum partial likelihood, model (1) is an extension of a proportional hazard Cox model.

The vector $\mathbf{z}_j$ has a multinomial distribution. Note that there are two levels of grouping. The first level is known, for example, healthcare providers as groups of patients) and we refer to these clusters as *groups*. The second level is the unknown clustering of groups (e.g. groups of healthcare providers with similar outcomes) that we want to detect, and we refer to these clusters as *latent populations*.

We assume that censoring is noninformative, thus that T_{ij}^* and C_{ij} are conditionally independent, given $\mathbf{X}_{ij}$, w_k and z_{jk}.

An application of this model to a real dataset is shown in Section 5 of Gasperoni *et al.* (2018).

In this case, the event of interest is the second admission in hospital of patients that suffer from Heart Failure (patients are grouped in healthcare providers).

Simulation of hierarchical time-to-event data with a latent clustering structure of groups

Two functions in the `discfrail` package simulate hierarchical time-to-event data in which there are clusters of groups with the same frailty.

The input parameters of the simulation algorithm are:

- J: the number of groups;
- N_j: the number of individuals in each group (N_j can be a *scalar* and in this case N_j is assumed to be equal in all groups; N_j can be a *vector*, J x 1, or it can be *NULL* in which case the N_j are drawn independently from a $Poisson(50)$ distibution.
- $\Lambda_0(t)$: the cumulative baseline hazard;
- $\boldsymbol{\pi}$: the K-dimensional probability vector of belonging to a single population;
- $\mathbf{w}$: the K-dimensional frailty vector;
- $\boldsymbol{\beta}$: the vector of regression parameters;
- $cens_{perc}$, the proportion of events which are censored.

Several steps are needed for building the dataset. The simulation procedure uses the method presented by Bender *et al.* (2005). We use two probability results to obtain Eq.(2): the inverse probability method and the fact that if a generic random variable V is distributed as $\mathcal{U}[0,1]$ then $1-V$ is still a $\mathcal{U}[0,1]$.

$$U_{ij} = 1 - F(t_{ij}; \mathbf{X}_{ij}, w) = \exp\{-\Lambda_0(T_{ij}) w_k \exp\{X_{ij}^T \beta\}\} \sim \mathcal{U}[0,1] \tag{2}$$

Then, we are able to compute the survival times with Eq.(3) by inverting Eq.(2).

$$T_{ij} = \Lambda_0^{-1}\left(\frac{-\log(U_{ij})}{w_k \exp\{\mathbf{X}_{ij}^T \beta\}}\right) \tag{3}$$

The main difference between equations (2), (3) and the ones showed in Bender *et al.* (2005) is the frailty term, w_k.

The covariates are randomly generated from a normal distribution, $\mathbf{X}_{ij} \overset{i.i.d.}{\sim} N(0,1)$.

Exploiting equations (2) and (3) we are able to simulate the *event* times T_{ij}. Then, we generate the *censoring* times according to a Normal distribution with standard deviation equal to one, independently from the T_{ij}. The censoring scheme is reproduced according to Wan (2017) so that $cens_{perc}$ of the simulated times are censored.

Finally, we obtain the status variable as: $\delta_{ij} = \mathbf{1}_{(T^*_{ij} \leq C_{ij})}$.

The package includes two general choices of baseline survival function:

- `sim_weibdf`, a parametric baseline, specifically a Weibull distribution characterised by two parameters λ and ρ, with cumulative hazard function $\Lambda_0(t) = \lambda \cdot t^{\rho}$.
- `sim_npdf`, a generic baseline, where the user can supply Λ_0^{-1} as an R function of time.

In the first case, equation (3) becomes:

$$T_{ij} = \left(\frac{-\log(U_{ij})}{w_k \cdot \lambda \exp\{\mathbf{X}_{ij}^T \beta\}} \right)^{1/\rho} \tag{4}$$

Example: Weibull baseline hazard

In the following example we simulate data with a Weibull baseline.

```r
library( discfrail )
```

```
## Loading required package: survival
```

```r
J <- 100       # total number of groups
N <- 40        # number of units per group
lambda <- 0.5 # Weibull scale parameter
rho <- 1.4     # Weibull shape parameter
beta <- c( 1.6, 0.4 ) # log hazard ratios for covariates
p <- c( 0.7, 0.3 )     # mixing proportions
w_values <- c( 1.2, 2.1 ) # frailty values
cens_perc <- 0.1   # percentage of censored events
set.seed(1200)
```

```r
data_weib <- sim_weibdf( J, N, lambda, rho, beta, p, w_values, cens_perc)
head( data_weib )
```

```
##   family      time status          x.1        x.2 belong
## 1      1 6.3524700      0 -0.848038891 -1.5494394    1.2
## 2      1 0.1524241      1 -0.047688486  1.8599481    1.2
## 3      1 0.7678551      1 -0.005320405  1.4354504    1.2
## 4      1 0.9151902      1  0.354866150  0.6519595    1.2
## 5      1 0.2059090      1  1.333647771  1.3741231    1.2
## 6      1 0.6141216      1  0.805291478  0.4975223    1.2
```

The simulated data object contains components that include

- `family`, the group indicator. In this case there are 40 individuals in each simulated group, and this can be checked here with `table(data_weib$family)`.
- `status`: the survival status δ_{ij}. Here we can check the proportion of censored events:

```r
table(data_weib$status) / sum(table(data_weib$status))
```

```
##
##       0       1
## 0.10025 0.89975
```

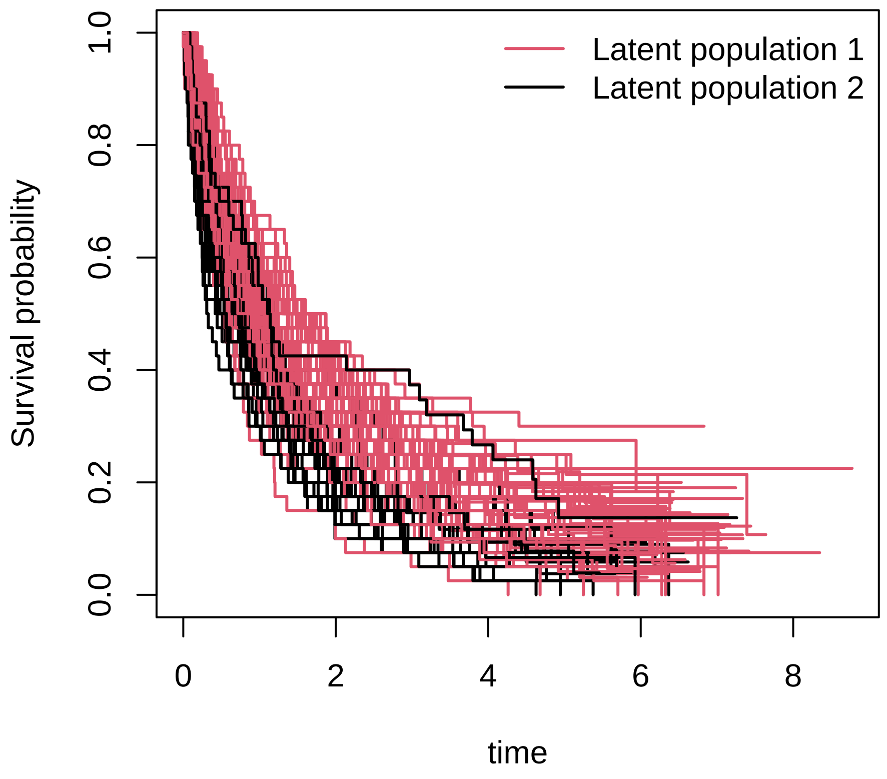
- `belong`: $\mathbf{w}$ the frailty values. Here we can check that the proportion of groups belonging to each latent population agrees with the values we specified for `p`:

```r
table(data_weib$belong) / sum(table(data_weib$belong))
```

```
##
## 1.2 2.1
## 0.7 0.3
```

We can visualize the simulated group-specific survivor curves by Kaplan-Meier estimation, as follows. The curves are coloured according to the latent population to which each group belongs.

```r
fit1 <- coxph( Surv( time, status ) ~ family, data_weib, method = 'breslow')
sfit1 <- survfit(Surv( time, status ) ~ family, data_weib)
lat_pop = rep( 2, J )
lat_pop[ which( data_weib$belong[ seq( 1, dim(data_weib)[1], N ) ] %in%
                  w_values[2]) ] = 1
plot( sfit1, col = lat_pop, xlab = 'time', ylab = 'Survival probability',
      lwd = 1.5 )
legend( 'topright', paste('Latent population', 1:2), col = 2:1,
        lty = rep(1,2), lwd = 1.5, bty = 'n' )
```



This illustrates the substantial heterogeneity between groups.

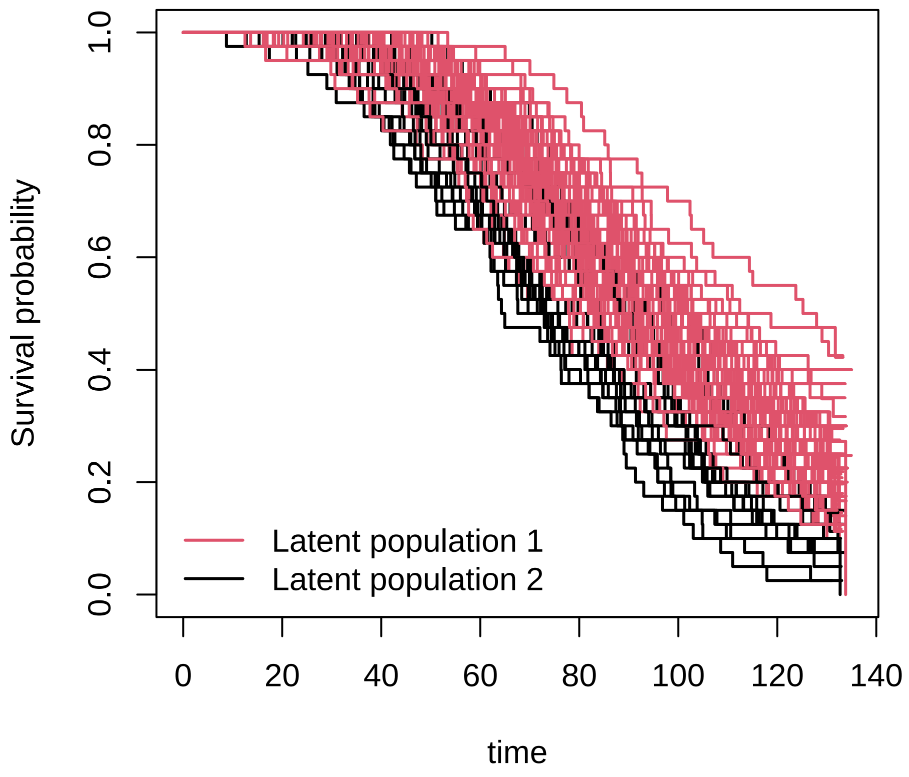
Example: Generic baseline hazard

The following code simulates the same form of data, but this time with a user-specified inverse cumulative hazard function $\Lambda_0(t)$ supplied as an R function.

```r
J <- 100
N <- 40
Lambda_0_inv = function( t, c=0.01, d=4.6 ) ( t^( 1/d ) )/c
beta <- 1.6
p <- c( 0.8, 0.2 )
w_values <- c( 0.8, 1.6 )
cens_perc <- 0.2
data_np <- sim_npdf( J, N, beta, Lambda_0_inv, p, w_values, cens_perc)
```

And again, the simulated survivor functions can be visualized as follows:

```r
fit1 <- coxph( Surv( time, status ) ~ family, data_np, method = 'breslow')
sfit1 <- survfit(Surv( time, status ) ~ family, data_np)
lat_pop = rep( 2, J )
lat_pop[ which( data_np$belong[ seq( 1, dim(data_np)[1], N ) ] %in%
                  w_values[2]) ] = 1
plot( sfit1, col = lat_pop, xlab = 'time', ylab = 'Survival probability',
      lwd = 1.5 )
legend( 'bottomleft', paste('Latent population', 1:2), col = 2:1,
        lty = rep(1,2), lwd = 1.5, bty = 'n' )
```



Cox model with a nonparametric discrete frailty

The function `npdf_cox` can be used to fit the Cox model with nonparametric discrete frailty. This is based on maximum partial likelihood using an EM algorithm – see Section 3 of Gasperoni *et al.* (2018). We illustrate the use of this function to fit this model to the dataset `data_weib` that was previously simulated.

The first argument specifies the outcome as a `Surv` object, and any covariates, in the standard R formula syntax used by `coxph` in the standard `survival` package. Another required argument is `groups`, which names a variable in the data containing the group membership indicators.

The number of latent populations can be specified in two alternative ways. If `npdf_cox` is called with `estK=FALSE`, then one model is fitted with the number of latent populations fixed at the number specified by the `K` argument to `npdf_cox`.

Alternatively, if `npdf_cox` is called with `estK=TRUE` (which is the default choice if `estK` is omitted) then multiple models are fitted with the number of latent groups ranging from 1 to `K`. This is done in the following example:

```
test_weib <- npdf_cox( Surv(time, status) ~ x.1 + x.2, groups = family,
                       data = data_weib, K = 4, eps_conv=10^-4)
```

Printing the fitted model object shows, firstly, the model comparison statistics for the models fitted. The estimates from the best-fitting model according to the preferred criterion (specified in the `criterion` argument to `npdf_cox`, by default, this is BIC) are then presented.

```
test_weib
```

```
##
## Call:
## npdf_cox(formula = Surv(time, status) ~ x.1 + x.2, groups = family,
##     data = data_weib, K = 4, eps_conv = 10^-4)
##
## Model comparison:
##   K K_fitted      llik      AIC      BIC
## 1 1        1 -28515.39 57036.79 57055.36
## 2 2        2 -28453.33 56916.66 56947.60
## 3 3        2 -28457.40 56928.80 56972.12
## 4 4        2 -28470.35 56958.69 57014.39
## Optimal K:
## Laird   AIC   BIC
##     2     2     2
##
## Best model according to BIC:
## Nonparametric discrete frailty Cox model fit with K=2 latent populations
##
```

```
## Estimated parameters and standard errors:
##          est seLouis seExact
## p1     0.694  0.0498  0.0499
## p2     0.306  0.0498  0.0499
## w2/w1  1.816  0.0693  0.0694
## x.1    1.640  0.0271  0.0271
## x.2    0.408  0.0169  0.0169
##
## Log-likelihood: -28453
## AIC: 56917
## BIC: 56948
## Fitted K: 2
##
## To examine other models, look at `models` component
```

According to AIC and BIC, the optimal $\hat{K}$ is equal to 2, which was the true value used to generate the data.

The criterion of Laird (1978) works as follows. At the end of the model fitting algorithm, each group is assigned to a latent population. The number of distinct "fitted" latent populations for each model is displayed in the `K_fitted` column of the model comparison table. Then the best-fitting model according to the Laird (1978) criterion is the maximum `K` among the fitted models for which `K=K_fitted`, that is, for which every latent population has at least one individual assigned to it. In this example, the optimal $\hat{K}$ is `4`.

The estimates $\hat{\boldsymbol{\pi}}$ are $[0.665, 0.335]$, close to the true values of $[0.7, 0.3]$. The estimated ratio $\hat{w}_2/\hat{w}_1 = 1.747$, again close to the true 1.75 (2.1/1.2). Exact standard errors and standard errors according to the method of Louis (1982) are also reported. See the Supplementary Material of Gasperoni et al.(2018) for more information about the computation of standard errors. Note that if `K_fitted` is less than `K` then the standard errors cannot be computed under these methods.

The function `npdf_cox` returns a list with the following components, illustrated for this example as follows:

- `test_weib$model` is a list with one element for each fitted model, in this case 4 elements. The i^{th} element of the list contains the estimates with i latent populations. We show the output related to $K = 3$.

```
test_weib$model[[3]]
```

```
##
## Nonparametric discrete frailty Cox model fit with K=3 latent populations
##
## Estimated parameters and standard errors:
##           est
## p1    0.00725
## p2    0.68663
## p3    0.30612
## w2/w1 1.06399
## w3/w1 1.93015
## x.1   1.63942
## x.2   0.40845
##
## Log-likelihood: -28457
## AIC: 56929
## BIC: 56972
## Fitted K: 2
```

- `test_weib$comparison` is a matrix in which the model comparison information, including the log-likelihood, AIC and BIC, is reported for each value of K.

```
test_weib$comparison
```

```
##   K K_fitted      llik      AIC      BIC
## 1 1        1 -28515.39 57036.79 57055.36
## 2 2        2 -28453.33 56916.66 56947.60
## 3 3         2 -28457.40 56928.80 56972.12
## 4 4         2 -28470.35 56958.69 57014.39
```

- `test_weib$Kopt` reports the optimal K according to Laird, AIC and BIC.

```
test_weib$Kopt
```

```
## Laird   AIC   BIC
##     2     2     2
```

- `test_weib$criterion` is the criterion used to choose the optimal K.

```
test_weib$criterion
```

```
## [1] "BIC"
```

- `test_weib$mf` is the *model frame*, the data frame used to fit the model.

Since the data are simulated, we are also able to check whether the groups are correctly assigned. We note that only 2 groups are misclassified (42 and 98).

```
# true distinct frailty for each group
w <- data_weib$belong[!duplicated(data_weib$family)]
# true latent population for each group
real_lat_pop <- match(w, sort(unique(w)))
table( test_weib$models[[2]]$belonging, real_lat_pop )
```

```
##    real_lat_pop
##       1  2
##    1 67  2
##    2  3 28
```

```
misc_ind = which( test_weib$models[[2]]$belonging != real_lat_pop )
misc_ind
```

```
## [1] 29 41 54 60 61
```

```
test_weib$models[[2]]$alpha[ misc_ind, ]
```

```
##            [,1]      [,2]
## [1,] 0.73278706 0.2672129
## [2,] 0.09290722 0.9070928
## [3,] 0.04250517 0.9574948
## [4,] 0.78118633 0.2188137
## [5,] 0.16067014 0.8393299
```

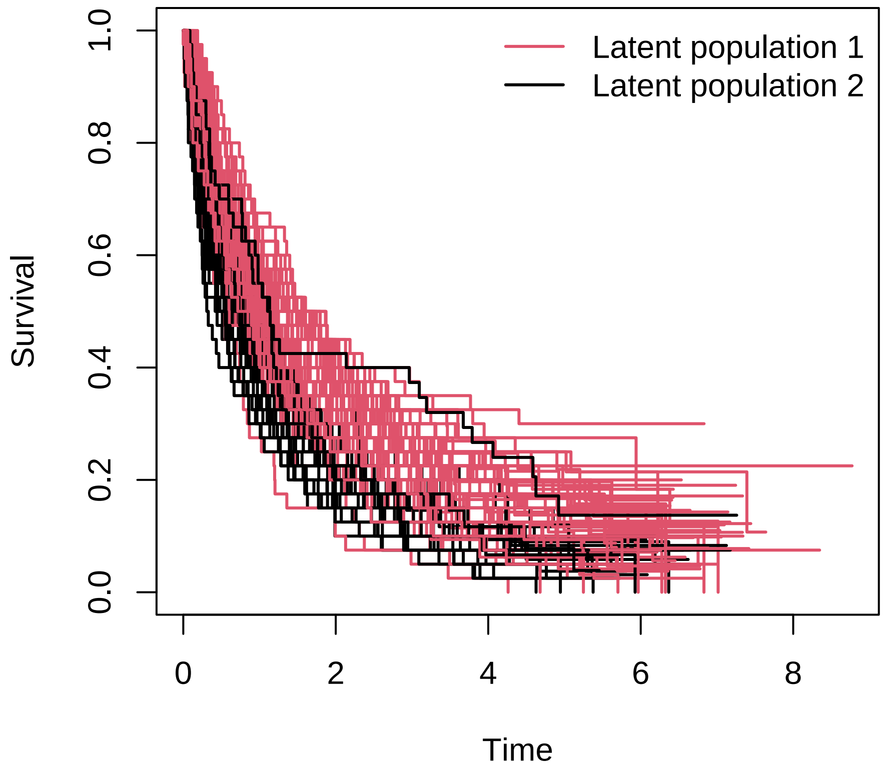
We can investigate the posterior probabilities of being part of a specific latent population (α_{jk}, see Section 3.1 of Gasperoni *et al.* (2018)). For some groups there is great uncertainty about the latent population to which they belong (i.e., group 98 is assigned to latent population 1 and 2 with probabilities of 0.66, 0.34 respectively). The following code identifies those groups that have probabilities between 0.05 and 0.95 of being assigned to latent population one and two (note that this list includes the two "misclassified" groups highlighted above).

```
#alpha matrix: alpha_{jk}
#is the probability that group j is assigned to latent population k.
#test_weib$models[[2]]$alpha
assign_not_sure1 = which( test_weib$models[[2]]$alpha[ ,1 ] < 0.95 &
                            test_weib$models[[2]]$alpha[ ,1 ] > 0.05 &
                            test_weib$models[[2]]$alpha[ ,2 ] < 0.95 &
                            test_weib$models[[2]]$alpha[ ,2 ] > 0.05 )
assign_not_sure1
```

```
##  [1]  7 10 24 29 41 46 56 60 61 82 95 97 99
```

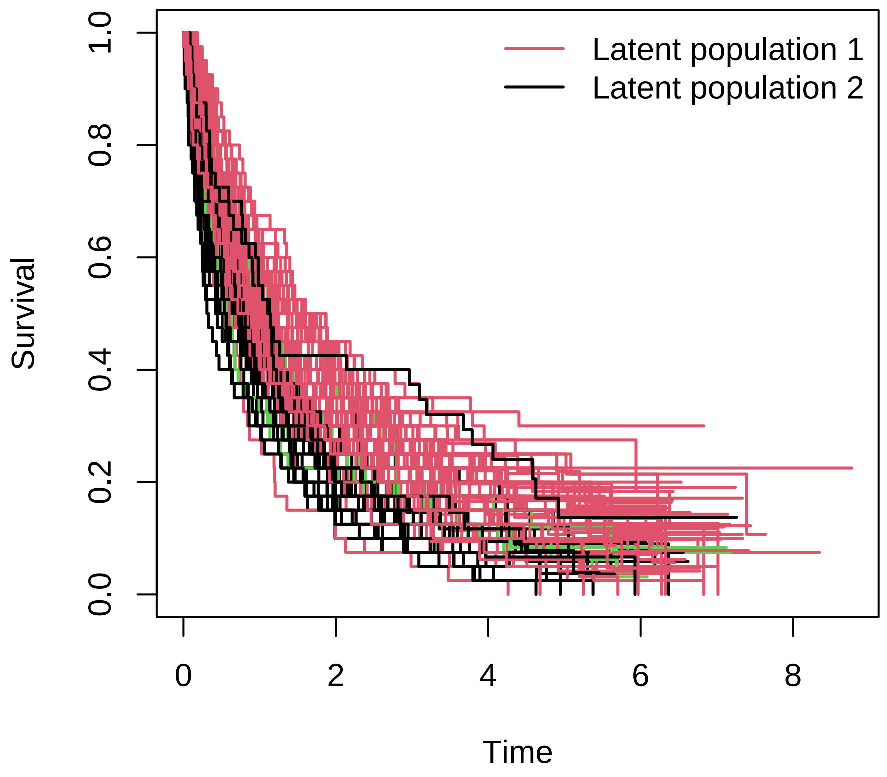
We can visualize the fitted survivor curves for each group through the `plot` method. The default, shown below, uses Kaplan-Meier estimates. It is also possible to plot Nelson-Aalen estimates (choosing `type = 'na'`). These plots automatically colour the group-specific curves according to the group's fitted latent population. This demonstrates how the group-specific frailties are clustered, with the higher-frailty population shown in black.

```
plot( test_weib, type = 'km', lwd = 1.5  )
legend( 'topright', paste( 'Latent population', c( 1, 2 ) ), lwd = 1.5,
        col = c( 2, 1 ), lty = rep( 1, 2 ), bty = 'n' )
```



The following plot highlights in green the survival curves for the three groups in the simulated data that are “misclassified” by the model.

```
colors_misc = test_weib$models[[2]]$belonging + 1
colors_misc[ colors_misc == 3 ] = 1
colors_misc[ test_weib$models[[2]]$belonging != real_lat_pop ] = 3
plot( test_weib, type = 'km', col = colors_misc, lwd = 1.5  )
legend( 'topright', paste( 'Latent population', c( 1, 2 ) ), lwd = 1.5,
        col = c( 2, 1 ), lty = rep( 1, 2 ), bty = 'n' )
```



References

Bender R, Augustin T, Blettner M. **2005**. Generating survival times to simulate Cox proportional hazards models. *Statistics in medicine* **24**: 1713–1723.

Gasperoni F, Ieva F, Paganoni AM, Jackson CH, Sharples LD. **2018**. Nonparametric frailty Cox models for hierarchical time-to-event data. *Biostatistics*.

Laird N. **1978**. Nonparametric maximum likelihood estimation of a mixing distribution. *Journal of the American Statistical Association* **73**: 805–811.

Louis TA. **1982**. Finding the observed information matrix when using the EM algorithm. *Journal of the Royal Statistical Society. Series B (Methodological)*: 226–233.

Wan F. **2017**. Simulating survival data with predefined censoring rates for proportional hazards models. *Statistics in medicine* **36**: 838–854.