R Implementation of a General Semi-parametric Shared Frailty Model

Google Summer of Code 2015 Proposal

John V. Monaco
PhD Student, Pace University
jmonaco@pace.edu, vmonaco.com

March 16, 2015

Abstract

Convenient and reliable means of survival analysis is critical to practitioners. Some tools exist for estimating the parameters of a shared frailty model, but they are currently incomplete and lagging behind the state-of-the-art. This work proposes an R package based on established modern research in estimating the parameters of a shared frailty model.

Contents

1 Introduction

Survival analysis is the study of time intervals. We may be interested in predicting the time until an event occurs, such as biological or mechanical failure, given some historical data. Or we may have to decide whether a covariate has any effect on the expected survival time of an individual. Several properties of survival data make survival analysis unique compared to other temporal analyses. Often, some of the data is censored, and we only know the time interval during which an event did not occur. This situation is common in clinical data, when a subject becomes unavailable for observation. Since the time of the event is not known, the interval is said to be *right censored*. Additionally, survival times generally come from skewed distributions [1].

In multivariate survival analysis, we may be faced with the task of deciding whether one or more covariates have an effect on the expected survival time. The covariates can be continuous or discrete. For example, in clinical data, it is common to consider age, gender, and smoking habits as potential indicators for disease risk or life expectancy. The decision whether to include any of these covariates must be statistically significant.

In dealing with clinical data, there are many scenarios where the survival times are clustered. This can be due to observations at various testing facilities (clustered by location), subjects that share a common gene, or some other predisposition that cannot be observed directly. Within each *family*, the survival times are independent, but as a whole, there exists a dependency between survival times and the *frailty* of each family. That is, an unobserved frailty value for a family may determine the survival times of those family members.

The Cox Proportional Hazards (PH) regression model was introduced several decades ago by Sir David Cox [2]. This model defines a *hazard function*, which is the risk of an event occurring at any given time as a function of the observed covariates. Unable to accurately model clustered data, a new model was proposed. The *shared frailty model* is an extension to the well-known Cox PH model, and introduces a multiplicative term in the hazard function for each family [6]. The proposed project is the implementation of a shared frailty model. This proposal presents the motivation and strategy to develop an R package capable of parameter estimation, data generation, and visualization, and is largely an implementation of the algorithm proposed in [3].

2 Motivation

Currently, parameters in a frailty model are generally determined by expectation-maximization (EM) [5]. In [3], an alternative algorithm is proposed that avoids several limitations of using EM, described in this section.

The hazard function in a shared frailty model is given by

$$\lambda_{ij}(t|\omega_i) = \lambda_0(t)\omega_i \exp(\beta' \mathbf{Z_{ij}}) \tag{1}$$

where t is time, ω_i is the frailty value for family i, $\lambda_0(t)$ is the baseline hazard function, β is the coefficient vector, and $\mathbf{Z_{ij}}$ is the vector of covariates for individual j in family i. The ω come from some distribution $f(\omega;\theta)$, where θ is a parameter vector. While only several frailty distributions have known solutions using EM, the estimation procedure defined in [3] is capable of handling general frailty distributions with finite moments. The implementation of a general shared frailty model will allow practitioners to choose the appropriate frailty distribution based on their data. Additionally, results obtained with the proposed package can be interpreted with confidence as this estimator has proved consistency and asymptotic normality .

Table 1: R packages for analyzing and generating survival data.

Package	Frailty distributions	Avg. weekly downloads
survival	gamma, log-normal, log-t	2003.1
coxme	log-normal	167.4
frailtypack	gamma, log-normal	80.3
MST	gamma, log-normal	57.6
phmm	log-normal	46
parfm	gamma, positive stable, inverse Gaussian	45.9
survBayes	gamma, log-normal	35.7

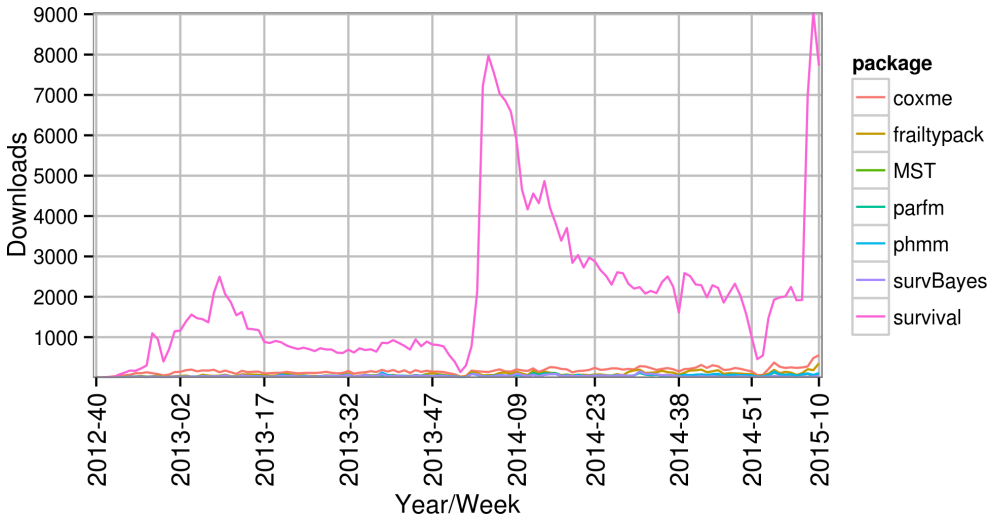

Figure 1: RStudio CRAN mirror weekly downloads of survival analysis packages

2.1 Survey of survival analysis software

There are currently several R packages available for the analysis and generation of survival data with a shared frailty model. Most of these packages are well documented, although it is not always clear which one should be used [4]. These are summarized in Table 1, with the supported frailty distributions and the average number of weekly downloads from the RStudio CRAN mirror, which has made its download logs available[1].

While some of these packages are able to generate survival data, such as MST, most do not have this capability. They either contain synthetic and/or real-world datasets or no data at all. The survsim package is dedicated to generating survival data, but it is not capable of generating data for all of the models in other packages. It is important for any generative model to be shipped with facilities for generating synthetic data, as this is a common procedure in model validation. In favor of self-contained functionality, the necessary data generation procedures will be developed as part of the proposed package.

To demonstrate the demand for survival analysis tools, the weekly downloads of the packages in Table 1 are shown in Figure 1. The survival package is by far the most popular, with increasingly larger spikes at the beginning of the year since 2012. The lesser-known packages also continue to grow in popularity, perhaps an indication that functionality outside the commonly used tools is desired. The proposed package is a vital addition to the battery of survival analysis tools available.

[1] An interesting dataset in its own right: `http://cran-logs.rstudio.com/`

2.2 Simulation

To demonstrate the utility of frailty models and the need for versatile parameter estimation procedures, several simulations are run. Data is generated according to the simulation described in [3], namely with survival function

$$S(t|\mathbf{Z},\omega) = \exp\left\{-\omega \exp(\beta'\mathbf{Z})(0.01t)^{4.6}\right\}$$

where ω is the frailty value shared for each family, $\mathbf{Z}$ is a vector of covariates, and β is the vector of covariate coefficients. First, for each family, a frailty value is sampled from either a gamma or log-normal distribution. Random covariates for each individual are generated from a standard normal or uniform distribution. The survival time for individual j in family i is generated by

$$t_{ij} = 0.01^{-1}\left(\frac{-\log u}{\omega_i \exp\beta'\mathbf{Z}}\right)^{4.6^{-1}}$$

where u is uniformly distributed between 0 and 1. Censorship times, c, are sampled from either $N(130, 15^2)$ or $N(60, 15^2)$, yielding approximately 35% and 85% censorship respectively. Let $\Delta = I\{t \leq c\}$ denote the occurrence of an event, where $I\{\cdot\} = 1$ if the condition is true and 0 otherwise. The observed data consist of $\{t, \Delta, \mathbf{Z}\}$. The resulting survival times are representative of a late onset disease, such as breast cancer [3]. Each simulation contains 300 families of size 2 and is repeated 500 times to obtain the empirical mean and standard deviation (SD). Source code for the simulation can be found in Appendix C.

The goal of this simulation is primarily to demonstrate the outcome of parameter estimation under the assumption of different frailty distributions. The frailty distribution parameter θ determines the strength of dependence between family members. Currently, practitioners are forced to choose from one of only several frailty distributions. Using the wrong frailty distribution can a serious bias into a model, as seen in Table 8 of the simulation results. For the gamma distribution, θ^{-1} is the shape and scale parameter

$$\omega \sim \Gamma(\frac{1}{\theta}, \frac{1}{\theta})$$

while for the log-normal, θ is the log variance, where the log-normal given by

$$f(\omega;\theta) = \frac{1}{\tau\sqrt{\theta 2\pi}} \exp\left(\frac{-(\ln\tau)^2}{2\theta}\right) \tag{2}$$

Note that this definition is slightly different from the log-normal traditionally defined with a log-standard deviation parameter. When $\theta = 2$, the log-normal has a heavier tail than the gamma, as shown in Figure 2. As θ increases, the data become more clustered with either distribution, and this effect is amplified with a log-normal frailty distribution. For detailed analysis of the simulation, see Appendix C.

3 Proposed work

There will be two deliverables upon completing the proposed project: an R package capable of estimating the parameters of a shared frailty model under a wide variety of distributions, and an article documenting the package to be submitted to the Journal of Statistical Software.

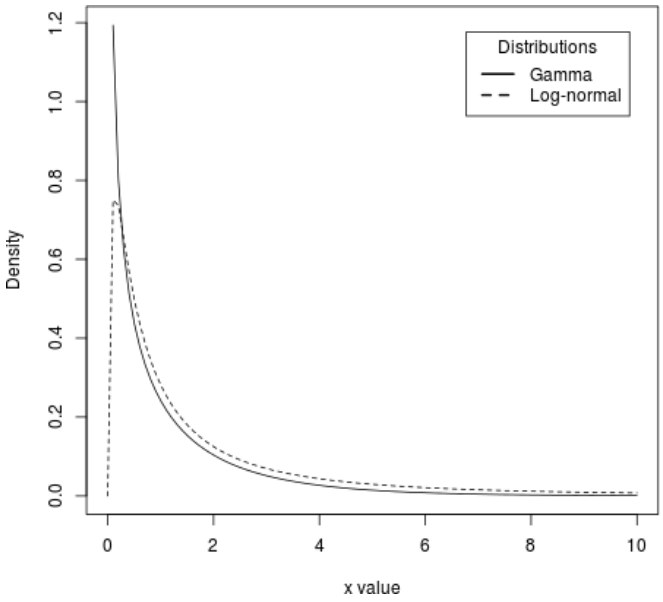

Figure 2: Gamma vs. log-normal for $\theta = 2$

3.1 Functionality

Most importantly, the package will provide a function capable of parameter estimation under the shared frailty model define in 1. Parameters to be estimated are the frailty distribution parameter θ, regression coefficient vector β, and baseline hazard function $\lambda_0(t)$. The assumptions of the model can be found in [7]. The parameter estimation function will be tested with well established datasets currently available in other R survival packages, such as the NCCTG Lung Cancer Data. Artificial data from a new function will also be used to verify the correctness of the implementation.

Functions to obtain confidence intervals via bootstrapping and resampling will be developed. This is import for model validation, so that the empirical and theoretical parameter variances can be compared. It also allows paves the way for Monte Carlo hypothesis testing.

The package will contain functions to generate synthetic data under a wide variety of conditions. This will enable practitioners to quickly determine the behavior of parameter estimation under various assumptions about the data. The data generation function will allow for control over the following:

- Frailty and covariate distributions
- Family size, censorship
- Constant scaling parameters

As an example of a data generation with some of the capabilities listed above, see the source code in Appendix C. Lastly, functions for visualization will be included where appropriate. The output will be similar to existing survival analysis models to minimize the learning curve for practitioners who wish to take advantage of this function. A summary of the proposed package capabilities is given in 2.

3.2 Timeline

Development will largely follow an incremental approach. The package design will reflect this, as the behavior of each function in the proposed package will be self-contained. This makes testing less complex. Much of the set

Table 2: Package functionality

Task	Description
Parameter estimation	Function to estimate the parameters of a shared frailty model
Data generation	Function to generate artificial data under a wide variety of conditions
Simulation	Functions for obtaining empirical confidence intervals and running simulations
Visualization	Functions for text-based and graphical model visualization

Table 3: Events timeline

Date	Milestone
6/7	Draft the data generation function. Behavior will be added as the parameter estimation function grows.
6/21	Implement the parameter estimation function with gamma and log-normal frailty. Verify results are consistent with theoretical values. This should be complete before midterm evaluation.
7/1	Add support for more frailty distributions: positive stable, inverse Gaussian, and power variance function (PVN)
7/7	I/O functions, mostly to be sure output is consistent with current R survival models
7/21	Functions for Monte Carlo hypothesis testing.
8/1	Code examples demonstrating function use cases and draft documentation.
8/7	Complete optimization. Runtime should be consistent with current implementations.
8/14	Pass R CMD Build and R CMD Check, submit package to CRAN.
8/21	Draft JSS article.

up and tear down in testing involves generating artificial data, which itself is a useful feature to have.

R provides comprehensive profiling tools, such as Rprof(), to help identify critical sections of code. I anticipate primarily computational bottlenecks, as the parameter estimation procedure requires several numerical integrations. It is unlikely that the function will require any space optimizations, as most survival datasets are relatively small.

The most difficult part of optimization comes with deciding the tradeoff between programming effort and program speed. To be adopted by practitioners, the parameter estimation should have a runtime proportional to current implementations. The computationally complexity of the algorithm in [3] was shown to be similar EM. To ensure the runtime is proportional to the existing EM implementations, critical sections of code will be compiled natively if necessary.

Guided by the official GSoC timeline, the proposed dates for milestone achievements are shown in 3. This table may change, subject to mentor feedback. I currently have no schedule conflicts during the work periods of the GSoC timeline.

4 Conclusions

There is no doubt a void in terms of survival analysis tools when it comes to using a general shared frailty distribution. To avoid the duplication of effort when a dataset warrants other frailty distributions besides what is currently available, an R package is proposed. The package will implement a proved-consistent and normal

estimator capable of handling general frailty distributions. The package will be accompanied by extensive documentation, allowing practitioners to immediately take advantage of this estimator, which currently has to open source implementation.

References

[1] TG Clark, MJ Bradburn, SB Love, and DG Altman. Survival analysis part i: basic concepts and first analyses. *British journal of cancer*, 89(2):232–238, 2003.

[2] David R Cox. Regression models and life-tables. In *Breakthroughs in Statistics*, pages 527–541. Springer, 1992.

[3] Malka Gorfine, David M Zucker, and Li Hsu. Prospective survival analysis with a general semiparametric shared frailty model: A pseudo full likelihood approach. *Biometrika*, 93(3):735–741, 2006.

[4] Katharina Hirsch and Andreas Wienke. Software for semiparametric shared gamma and log-normal frailty models: An overview. *Computer methods and programs in biomedicine*, 107(3):582–597, 2012.

[5] Samuli Ripatti and Juni Palmgren. Estimation of multivariate frailty models using penalized partial likelihood. *Biometrics*, 56(4):1016–1022, 2000.

[6] Terry M Therneau and Patricia M Grambsch. *Modeling survival data: extending the Cox model*. Springer Science & Business Media, 2000.

[7] David M Zucker, Malka Gorfine, and Li Hsu. Pseudo-full likelihood estimation for prospective survival analysis with a general semiparametric shared frailty model: Asymptotic theory. *Journal of Statistical Planning and Inference*, 138(7):1998–2016, 2008.

A About me

I am Computer Science PhD student at Pace University, working on a dissertation in behavioral biometrics. My research focuses on the use of timestamped events for human identification, verification, and prediction. I'm funded by the Department of Defense and attend school under the Information Assurance Scholarship Program (IASP).

I participated in GSoC 2013 as a graduate student, during which I proposed and developed a Moodle plugin for instructors to verify the identity of students taking online exams by their keystroke signatures. This project involved capturing key event times on the client side and an algorithm to extract features and classify on the server side. The project was completed successfully on time, and several of my publications have been base on that work. More recently, I participated in an international eye-movement biometric competition and came in first place. This work involved optimizing the multivariate Wald-Wolfowitz test, which I used in a distance-based classifier.

My goal in writing this proposal and working on this project is to develop a deeper understanding of multivariate survival analysis. I believe that the analysis of clustered survival times, or alternatively, repeating events from one individual, is intimately related to time-interval behavioral biometrics. I hope that others will benefit from this work, which a the survival analysis tools currently available.

B Simulation results

Simulations were performed with R version 3.1.3 running in an Ubuntu 12.04 desktop with AMD Phenom 9550 Quad-Core 2.20 GHz Processor. All of the simulation results are obtained using penalized partial likelihood estimation, namely the coxph function in the survival package with the Breslow method. Each simulation was repeated 500 times to obtain the empirical mean and SD. The empirical censorship and runtime (in seconds) are also reported. For more information, see Section 2.2.

Table 4: Ignoring the clustered data: A constant frailty is assumed when the actual frailty is either gamma or log-normal. The coefficients are consistently biased, as this model ignores the clustered data and underestimates the effect of the coefficient. The bias is less severe with greater censorship, likely the result of decreased clustering. There is more diversity at 80% censorship due to fewer individual from the same family being selected, though this conjecture has not been verified.

(a) Gamma frailty

$\beta^0 = \log 2$, $N(130, 15^2)$ censoring

	$\hat{\beta}$	Censoring	Runtime
Emp. mean	0.342	0.379	0.017
Emp. SD	0.056	0.024	0.001

$\beta^0 = \log 2$, $N(60, 15^2)$ censoring

	$\hat{\beta}$	Censoring	Runtime
Emp. mean	0.565	0.883	0.016
Emp. SD	0.128	0.013	0.001

$\beta^0 = \log 3$, $N(130, 15^2)$ censoring

	$\hat{\beta}$	Censoring	Runtime
Emp. mean	0.541	0.390	0.017
Emp. SD	0.061	0.025	0.001

$\beta^0 = \log 3$, $N(60, 15^2)$ censoring

	$\hat{\beta}$	Censoring	Runtime
Emp. mean	0.846	0.868	0.016
Emp. SD	0.127	0.014	0.001

(b) Log-normal frailty

$\beta^0 = \log 2$, $N(130, 15^2)$ censoring

	$\hat{\beta}$	Censoring	Runtime
Emp. mean	0.393	0.199	0.016
Emp. SD	0.049	0.019	0.001

$\beta^0 = \log 2$, $N(60, 15^2)$ censoring

	$\hat{\beta}$	Censoring	Runtime
Emp. mean	0.507	0.796	0.016
Emp. SD	0.094	0.018	0.001

$\beta^0 = \log 3$, $N(130, 15^2)$ censoring

	$\hat{\beta}$	Censoring	Runtime
Emp. mean	0.630	0.219	0.017
Emp. SD	0.056	0.019	0.001

$\beta^0 = \log 3$, $N(60, 15^2)$ censoring

	$\hat{\beta}$	Censoring	Runtime
Emp. mean	0.787	0.781	0.016
Emp. SD	0.091	0.019	0.001

Table 5: Gamma frailty with a single covariate: The model behaves as anticipated, and correctly estimates the coefficient and frailty distribution variance within error. One confound is the the SD for $\mathbf{Z} \sim \mathcal{U}(0,1)$ is closer to that found in [3], yet their simulation used covariates from a standard normal.

(a) Standard normal covariate,

$\beta^0 = \log 2$, $N(130, 15^2)$ censoring

	$\hat{\beta}$	$\hat{\theta}$	Censoring	Runtime
Emp. mean	0.686	1.977	0.379	0.199
Emp. SD	0.086	0.276	0.024	0.018

$\beta^0 = \log 2$, $N(60, 15^2)$ censoring

	$\hat{\beta}$	$\hat{\theta}$	Censoring	Runtime
Emp. mean	0.691	1.802	0.883	0.061
Emp. SD	0.169	1.011	0.013	0.009

$\beta^0 = \log 3$, $N(130, 15^2)$ censoring

	$\hat{\beta}$	$\hat{\theta}$	Censoring	Runtime
Emp. mean	1.090	1.978	0.390	0.191
Emp. SD	0.102	0.277	0.025	0.017

$\beta^0 = \log 3$, $N(60, 15^2)$ censoring

	$\hat{\beta}$	$\hat{\theta}$	Censoring	Runtime
Emp. mean	1.090	1.879	0.868	0.066
Emp. SD	0.187	0.921	0.014	0.009

(b) Uniform covariate

$\beta^0 = \log 2$, $N(130, 15^2)$ censoring

	$\hat{\beta}$	$\hat{\theta}$	Censoring	Runtime
Emp. mean	0.699	1.961	0.320	0.229
Emp. SD	0.246	0.274	0.024	0.020

$\beta^0 = \log 2$, $N(60, 15^2)$ censoring

	$\hat{\beta}$	$\hat{\theta}$	Censoring	Runtime
Emp. mean	0.746	1.837	0.863	0.063
Emp. SD	0.443	0.866	0.015	0.009

$\beta^0 = \log 3$, $N(130, 15^2)$ censoring

	$\hat{\beta}$	$\hat{\theta}$	Censoring	Runtime
Emp. mean	1.101	1.959	0.294	0.244
Emp. SD	0.248	0.265	0.024	0.020

$\beta^0 = \log 3$, $N(60, 15^2)$ censoring

	$\hat{\beta}$	$\hat{\theta}$	Censoring	Runtime
Emp. mean	1.146	1.843	0.841	0.069
Emp. SD	0.430	0.729	0.016	0.010

Table 6: Gamma frailty with two covariates: Again, the coefficients and frailty variance are correctly estimated within error.

(a) Standard normal covariate

$\beta_0 = \log 2$, $\beta_1 = \log 3$, $N(130, 15^2)$ censoring

	$\hat{\beta}_0$	$\hat{\beta}_1$	$\hat{\theta}$	Censoring	Runtime
Emp. mean	0.692	1.099	1.959	0.395	0.264
Emp. SD	0.086	0.100	0.281	0.024	0.021

$\beta^0 = \log 2$, $\beta_1 = \log 3$, $N(60, 15^2)$ censoring

	$\hat{\beta}_0$	$\hat{\beta}_1$	$\hat{\theta}$	Censoring	Runtime
Emp. mean	0.687	1.100	1.857	0.860	0.082
Emp. SD	0.161	0.176	0.873	0.015	0.013

Table 7: Log-normal frailty with a single covariate: With $\theta = 2$, the log-normal has a heavier tail than the gamma, and thus more clustering. As a result, with high censorship, the coefficients and frailty variance tend to by slightly underestimated. Similar to the gamma frailty, the error for uniformly-distributed individual covariates is higher than standard normal individual covariates.

(a) Standard normal covariate

$\beta^0 = \log 2$, $N(130, 15^2)$ censoring

	$\hat{\beta}$	$\hat{\theta}$	Censoring	Runtime
Emp. mean	0.671	1.808	0.199	0.196
Emp. SD	0.071	0.296	0.019	0.014

$\beta^0 = \log 2$, $N(60, 15^2)$ censoring

	$\hat{\beta}$	$\hat{\theta}$	Censoring	Runtime
Emp. mean	0.600	1.130	0.796	0.054
Emp. SD	0.105	0.265	0.018	0.006

$\beta^0 = \log 3$, $N(130, 15^2)$ censoring

	$\hat{\beta}$	$\hat{\theta}$	Censoring	Runtime
Emp. mean	1.061	1.802	0.219	0.184
Emp. SD	0.085	0.292	0.019	0.012

$\beta^0 = \log 3$, $N(60, 15^2)$ censoring

	$\hat{\beta}$	$\hat{\theta}$	Censoring	Runtime
Emp. mean	0.953	1.162	0.781	0.058
Emp. SD	0.106	0.269	0.019	0.006

(b) Uniform covariate

$\beta^0 = \log 2$, $N(130, 15^2)$ censoring

	$\hat{\beta}$	$\hat{\theta}$	Censoring	Runtime
Emp. mean	0.683	1.889	0.141	0.223
Emp. SD	0.224	0.296	0.017	0.016

$\beta^0 = \log 2$, $N(60, 15^2)$ censoring

	$\hat{\beta}$	$\hat{\theta}$	Censoring	Runtime
Emp. mean	0.630	1.164	0.763	0.062
Emp. SD	0.347	0.253	0.020	0.007

$\beta^0 = \log 3$, $N(130, 15^2)$ censoring

	$\hat{\beta}$	$\hat{\theta}$	Censoring	Runtime
Emp. mean	1.084	1.922	0.120	0.234
Emp. SD	0.227	0.304	0.015	0.018

$\beta^0 = \log 3$, $N(60, 15^2)$ censoring

	$\hat{\beta}$	$\hat{\theta}$	Censoring	Runtime
Emp. mean	0.989	1.199	0.733	0.068
Emp. SD	0.331	0.252	0.021	0.008

Table 8: Assuming the wrong frailty: When the true frailty is log-normal and we assume a gamma distribution, the coefficients are underestimated. This effect is pronounced for higher censorship. This is a result of underestimating how much the data are clustered, since the log-normal has a heavier tail and produces more clustered data. Underestimation is not as severe when the true frailty is gamma and a log-normal is assumed. In this case, we assume the data is more clustered that it actually is.

(a) Gamma frailty, assume log-normal

$\beta^0 = \log 2$, $N(130, 15^2)$ censoring

	$\hat{\beta}$	$\hat{\theta}$	Censoring	Runtime
Emp. mean	0.615	2.050	0.379	0.139
Emp. SD	0.077	0.291	0.024	0.012

$\beta^0 = \log 2$, $N(60, 15^2)$ censoring

	$\hat{\beta}$	$\hat{\theta}$	Censoring	Runtime
Emp. mean	0.620	0.866	0.883	0.041
Emp. SD	0.140	0.304	0.013	0.003

$\beta^0 = \log 3$, $N(130, 15^2)$ censoring

	$\hat{\beta}$	$\hat{\theta}$	Censoring	Runtime
Emp. mean	0.979	2.111	0.390	0.136
Emp. SD	0.091	0.304	0.025	0.011

$\beta^0 = \log 3$, $N(60, 15^2)$ censoring

	$\hat{\beta}$	$\hat{\theta}$	Censoring	Runtime
Emp. mean	0.962	0.962	0.868	0.044
Emp. SD	0.143	0.287	0.014	0.004

(b) Log-normal frailty, assume gamma

$\beta^0 = \log 2$, $N(130, 15^2)$ censoring

	$\hat{\beta}$	$\hat{\theta}$	Censoring	Runtime
Emp. mean	0.665	1.188	0.199	0.213
Emp. SD	0.073	0.249	0.019	0.021

$\beta^0 = \log 2$, $N(60, 15^2)$ censoring

	$\hat{\beta}$	$\hat{\theta}$	Censoring	Runtime
Emp. mean	0.671	1.891	0.796	0.079
Emp. SD	0.121	0.653	0.018	0.012

$\beta^0 = \log 3$, $N(130, 15^2)$ censoring

	$\hat{\beta}$	$\hat{\theta}$	Censoring	Runtime
Emp. mean	1.044	1.149	0.219	0.197
Emp. SD	0.085	0.168	0.019	0.016

$\beta^0 = \log 3$, $N(60, 15^2)$ censoring

	$\hat{\beta}$	$\hat{\theta}$	Censoring	Runtime
Emp. mean	1.053	1.693	0.781	0.081
Emp. SD	0.129	0.562	0.019	0.011

C Simulation source code

This function generates clustered survival data similar to that in [3]. For the complete source code, see `https://github.com/vmonaco/gsoc-survival`

```
survivalgen <- function(beta = c(log(2)), # Covariate coefficients
                        frailty = c("gamma", "log.normal"), # Frailty distribution
                        censor.mu = 130, # Gaussian distribution for censoring
                        censor.sigma = 15,
                        theta = 2, # Frailty distribution parameter
                        N = 300, K = 2, # Number of families and family size
                        tau = 4.6,
                        C = 0.01,
                        covariates = c("normal", "uniform")
  )
{
  # Generate the covariates
  p <- length(beta)
  Z <- matrix(0, nrow=N*K, ncol=p)
  for (j in 1:p) {
    if (covariates=="normal") {
      Z[, j] <- rnorm(N*K)
    } else if (covariates=="uniform") {
      Z[, j] <- runif(N*K)
    }
  }
  betaZ <- Z%*%beta

  # Generate the frailty for each family
  if (frailty=="gamma") {
    w <- 1
    if ( theta != 0 ) { w <- rep(rgamma(N, 1/theta, 1/theta), rep(K, N)) }
    rate <- exp(betaZ)*w
  } else if (frailty=="log.normal") {
    w <- 1
    if ( theta != 0 ) { w <- rep(rnorm(N, 0, sqrt(theta)), rep(K, N)) }
    rate <- exp(betaZ + w)
  }

  # Survival time is similar to that of a late onset disease with default params
  u <- runif(N*K)
  obs.time <- ((-log(u)/rate)^(1/tau))/C

  # Non-informative right censoring
  censor.time <- rnorm(N*K, censor.mu, censor.sigma)
  obs.status <- sign(obs.time <= censor.time)
  obs.time <- pmin(obs.time, censor.time)

  colnames(Z) <- paste("Z", 1:p, sep="")
  dat <- data.frame(family=rep(1:N, rep(K,N)),
                    rep=rep(1:K, N),
                    time=obs.time,
                    status=obs.status,
                    Z)
  return(dat)
}
```