Letter to the Editor

Comment on Moriña and Navarro (2017)

Maria Teresa Giraudo *[1], Fulvio Ricceri[2]

and Elena Rosso[1]

[1]*Department of Mathematics "Giuseppe Peano", University of Turin, Via Carlo Alberto 8, Turin, Italy*

[2]*Department of Clinical and Biological Sciences, University of Turin, Regione Gonzole 10, Orbassano (TO), Italy and Unit of Epidemiology, Regional Health Service ASL TO3, Via Sabaudia 164, Grugliasco (TO), Italy*

Keywords

Competing risks; Simulation; R; `survsim`; Survival probability

This letter intends to comment on some aspects of the software described in the article Moriña and Navarro (2017). The goal of that work, as specified in the Abstract,

*Corresponding author: mariateresa.giraudo@unito.it

was to introduce the latest version published at the time of writing the article of the R package `survsim`, in which new functions were added allowing to simulate a cohort in a competing risks setting using cause-specific hazards models.
The package `survsim` had already been introduced by the same authors in Moriña and Navarro (2014), where its use in simple survival analysis, in recurrent events survival analysis, and in multiple events survival analysis is described in detail. The main novelty of the version at the time (1.1.4, available on CRAN) consisted in the fact that it allowed to simulate survival data also in the presence of competing risks by means of the function *crisk.sim*, with a structure otherwise similar to that of the previous functions. It is possible to simulate cause-specific hazards according to Weibull distribution, of which exponential is a particular case, log-normal distribution, or log-logistic distribution. Moreover, a random effect can be introduced in order to take individual heterogeneity into account and one can also assign covariate values to the simulated cohort individuals. The possible distributions for covariates are normal, uniform, and Bernoulli distributions. To simulate the data for each individual, the function *crisk.sim* calls the further internal function *crisk.ncens.sim*. We used the functions in the version 1.1.5 of the package `survsim`, available on CRAN from May 17, 2018, to simulate survival data for the study of a methodological problem related to competing risk analysis. In this context, we found an error in the implementation of the hazard function $h_W(t)$ for the Weibull distribution and a limitation in the simulation of censored subjects within the function *crisk.ncens.sim*.

As reported in Moriña and Navarro (2017), the formula of the hazard function for the Weibull distribution should be

$$h_W(t) = \lambda_j p t^{p-1}, \tag{1}$$

with the parametrization $\lambda_j = e^{-pt(\beta_0 + X_j\beta_j)}$ and where all terms are defined as explained in the cited paper. In the code, however, the exponent p is incorrectly assigned also to the term p of the product in (1) (i.e. the implemented hazard is $\lambda_j p^p t^{p-1}$).

The error has not been corrected even in the latest version of the software (1.1.6, available on CRAN from May 14, 2021). When the Weibull distribution is selected for the cause-specific hazards in the context of competitive risks studies, this leads to unreliable simulations of the all cause time to event density.

To illustrate this problem, we have simulated a cohort of $n = 100000$ subjects in a framework with two competing causes that they can suffer. We did not add censoring since it was not relevant in this context, while we considered one Bernoulli covariate and we chose the following set of parameters for the two Weibull distributions of the times to event:

- cause 1: $\begin{cases} p = 2 \\ \beta_0 = 7.65 \\ \beta_1 = -0.11 \end{cases}$

- cause 2: $\begin{cases} p = 2 \\ \beta_0 = 7.33 \\ \beta_1 = -0.66 \end{cases}$

In both cases the follow-up time was fixed at 10 years, corresponding to 3650 days. In Figure 1 we compare the theoretical all cause time to event densities corresponding to the subcohort with and without covariate effect (continuous lines, right and left panel respectively) with the simulated densities obtained by using the original code in *crisk.ncens.sim* (dotted lines) and by using a corrected version of the code (dashed lines). It is evident from the plots that the original code does not produce the correct density in either cases, as opposite to the version that we implemented.

[Figure 1 near here]

Moreover, we noticed that the simulation procedure of times to event implemented in the function *crisk.ncens.sim* does not produce subjects withdrawn alive from the study. Indeed, an event is simulated for a subject when the inversion algorithm finds a root for the distribution function of the all cause time to event inside the interval from the starting epoch to the follow up time. If this does not happen and the simulated censoring time lies outside the follow-up period, the individual is assigned all the same the status of event occurred. In this case the time to event coincides with the follow-up duration and a cause is chosen between the two possible ones (if two competing events are considered) according to the usual binomial experiment. In this way the proportion of individuals that have not experienced any

of the competing events is always null. This is a limitation as patients can have a high chance of surviving beyond the study period.
To illustrate this problem, we used the simulation procedure to estimate the probability of survival at 1, 5 and 10 years in the same situation of competing risks described above. In Table 1 we report the estimates of the survival probabilities obtained by using the original not corrected code in *crisk.ncens.sim* and by using a corrected version of the code and we compare them with the theoretical values. Since the error in the computation of the Weibull hazard is still present here, in any case the original program would not give a correct estimation of the survival, but the returned value is in any case always null.

[Table 1 near here]

We conclude this Letter by pointing out that both the error in the implementation of the hazard function for the Weibull distribution and the limitation in the event simulation routine inside the function *crisk.ncens.sim* could be remedied with small corrections to the function code. This would allow an efficient use of the whole package *survsim*, which is a useful tool for all those involved in research in the field of survival analysis in general and of competing risks in particular. A proposal for the update of the function, called in this version *new.crisk.ncens.sim*, is shown in the Appendix hereon.

References

[1] Moriña D. and Navarro A. Competing risks simulation with the survsim R package. *Communications in Statistics - Simulation and Computation* 2017; 46(7): 5712-5722.

[2] Moriña D. and Navarro A. The R package survsim for the simulation of simple and complex survival data. *Journal of Statistical Software* 2014; 59(2): 1-20.

Appendix

```
new.crisk.ncens.sim<-function(foltime, anc.ev, beta0.ev, anc.cens, beta0.cens,
    z = NULL, beta = 0, eff = 0, dist.ev, dist.cens, i, nsit)
{
  nid <- NA
  start <- NA
  stop <- NA
  obs <- NA
  time <- NA
  pro <- vector()
  cause <- NA
  a.ev <- vector()
  b.ev <- vector()
  a.cens <- NA
  b.cens <- NA
  obs[1] <- 1
  k.ev <- 1
  sum <- 0
  cshaz <- list()
  az1 <- vector()
  it<-0
  if (is.null(z)) {
    for (j in 1:nsit) {
      az1[j] <- 1
    }
  }
  else {
    for (j in 1:length(z)) {
```

```
    if (!is.na(z[[j]][1]) && z[[j]][1] == "gamma")
      az1[j] <- rgamma(1, as.numeric(z[[j]][2]), as.numeric(z[[j]][3]))
    if (!is.na(z[[j]][1]) && z[[j]][1] == "exp")
      az1[j] <- rgamma(1, 1, as.numeric(z[[j]][2]))
    if (!is.na(z[[j]][1]) && z[[j]][1] == "weibull")
      az1[j] <- rweibull(1, as.numeric(z[[j]][2]),
                         as.numeric(z[[j]][3]))
    if (!is.na(z[[j]][1]) && z[[j]][1] == "unif")
      az1[j] <- runif(1, as.numeric(z[[j]][2]), as.numeric(z[[j]][3]))
    if (!is.na(z[[j]][1]) && z[[j]][1] == "invgauss")
      az1[j] <- rinvgauss(1, as.numeric(z[[j]][2]),
                          as.numeric(z[[j]][3]))
  }
  if (length(z) == 1) {
    for (j in 2:nsit) {
      az1[j] <- az1[j]
    }
  }
}
if (dist.cens == "llogistic") {
  tc <- exp(rlogis(1, beta0.cens, anc.cens))
}
else {
  if (dist.cens == "weibull") {
    a.cens <- anc.cens
    b.cens <- (1/exp(-anc.cens * (beta0.cens)))^(1/anc.cens)
    tc <- rweibull(1, a.cens, b.cens)
  }
  else {
    if (dist.cens == "lnorm") {
      tc <- rlnorm(1, beta0.cens, anc.cens)
    }
    else {
      if (dist.cens == "unif") {
        tc <- runif(1, beta0.cens, anc.cens)
      }
    }
  }
}
suma <- vector()
for (m2 in 1:nsit) {
  suma[m2] <- 0
  for (m1 in 1:length(beta)) {
    suma[m2] <- suma[m2] + beta[[m1]][m2] * eff[m1]
  }
}
if (all(is.na(suma)))
  suma <- rep(0, nsit)
for (k in 1:nsit) {
  if (dist.ev[k] == "llogistic") {
    a.ev[k] <- 1/exp(beta0.ev[k] + suma[k])
    b.ev[k] <- anc.ev[k]
    cshaz[[k]] <- function(t, r) {
      par1 <- eval(parse(text = "a.ev[r]"))
      par2 <- eval(parse(text = "b.ev[r]"))
      z <- eval(parse(text = "az1[r]"))
      return(z * (par1 * par2 * (t^(par2 - 1)))/(1 +
               par1 * (t^par2)))
    }
  }
  else {
    if (dist.ev[k] == "weibull") {
      a.ev[k] <- beta0.ev[k] + suma[k]
      b.ev[k] <- anc.ev[k]
      cshaz[[k]] <- function(t, r) {
        par1 <- eval(parse(text = "a.ev[r]"))
        par2 <- eval(parse(text = "b.ev[r]"))
        z <- eval(parse(text = "az1[r]"))
        return(z * ((1/par2)/((exp(par1))^(1/par2)) )*
                 t^((1/par2) - 1))
      }
    }
    else {
      if (dist.ev[k] == "lnorm") {
        a.ev[k] <- beta0.ev[k] + suma[k]
        b.ev[k] <- anc.ev[k]
        cshaz[[k]] <- function(t, r) {
          par1 <- eval(parse(text = "a.ev[r]"))
          par2 <- eval(parse(text = "b.ev[r]"))
          z <- eval(parse(text = "az1[r]"))
          return(z * (dnorm((log(t) - par1)/par2)/(par2 *
                  t * (1 - pnorm((log(t) - par1)/par2)))))
        }
      }
    }
  }
}
A <- function(t, y) {
  res <- 0
  for (k in 1:length(cshaz)) {
    res <- res + integrate(cshaz[[k]], lower = 0.00001,
                           upper = t, r = k, subdivisions = 2000)$value
  }
  res <- res + y
  return(res[1])
}
u <- runif(1)
if (A(0.001, log(1 - u)) * A(foltime, log(1 - u)) > 0) {
  tb <- foltime
  cause <- 0
}
else
{tb <- uniroot(A, c(0, foltime), tol = 1e-05, y = log(1 - u))$root
it<-1
sumprob <- 0
for (k in 1:length(cshaz)) {
  sumprob <- sumprob + cshaz[[k]](tb, k)
}
for (k in 1:length(cshaz)) {
  pro[k] <- cshaz[[k]](tb, k)/sumprob
}
cause1 <- rmultinom(1, 1, prob = pro)
```

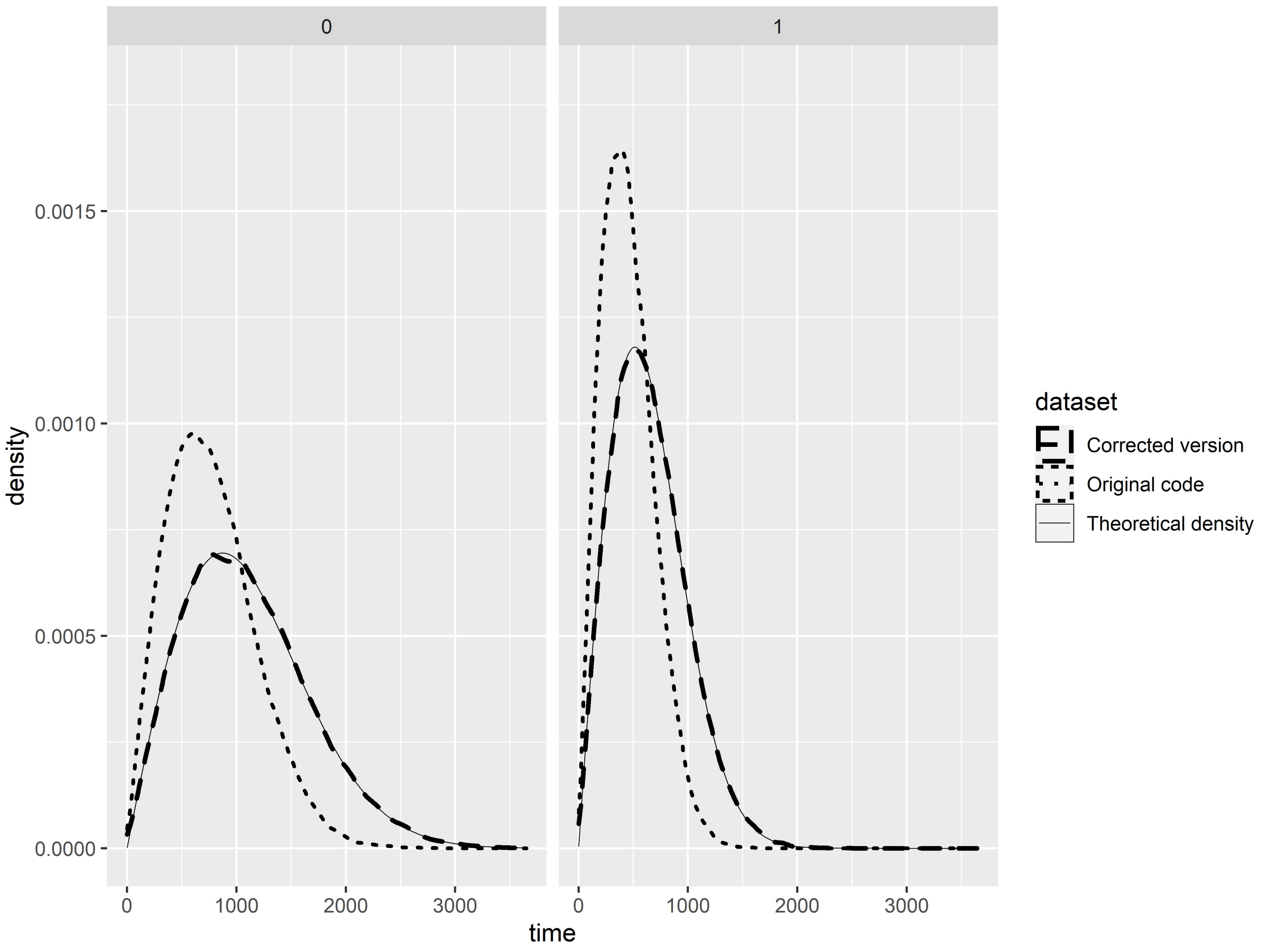
```
 for (k in 1:length(cshaz)) {
   if (cause1[k] == 1)
     cause <- k
 }}
 az <- az1[k]
 nid <- i


 if (tc < tb ) {
   tb <- tc
   it <- 0
   cause<-0
 }

 sim.ind <- data.frame(nid = nid, cause = cause, status = it, time= tb, z = az)
 for (k in 1:length(eff)) {
   sim.ind <- cbind(sim.ind, x = eff[k])
 }
 return(sim.ind)
}
```

Covariate	Follow-up (years)	Original code	Corrected version	Theoretical value
	10	0	0.0002402	0.0001593
0	5	0	0.1144569	0.1123423
	1	0	0.9163347	0.9162663
	10	0	0	1.142e-11
1	5	0	0.0015476	0.0018383
	1	0	0.7781463	0.7772777

Table 1: Survival probabilities

Figure captions

- Figure 1: Simulated densities (dotted lines: original code; dashed lines: corrected version) compared with the theoretical ones (continuous lines)


0
1
density
0.0015
0.0010
0.0005
0.0000
0
1000
2000
3000
time
dataset
Corrected version
Original code
Theoretical density