# Software for the algorithms used in “Nonparametric Estimation under Shape Constraints"

**Piet Groeneboom**

**Abstract:** We discuss software, written in `C` and used in Groeneboom and Jongbloed (2014), which we make available on this site, using `Rcpp`, see, e.g., Eddelbuettel (2013). Parallel to this development, we also create GUI (Graphical User Interface) applications for Mac OS 10.10 (Yosemite). The software can be used to reproduce the results and pictures in Groeneboom and Jongbloed (2014).

**AMS 2000 subject classifications:** Primary 62G05, 62N01; secondary 62-04.
**Keywords and phrases:** MLE, SMLE, iterative convex mnorant algorithm, Rcpp.

## 1. Introduction

In Groeneboom and Jongbloed (2014) frequent use is made of results of `C` programs, which implement algorithms for computing estimators iteratively, or for computing test statistics using bootstrap procedures. Unfortunately, these computer programs were written using standard routines from Numerical Recipes in `C` (see Press et al. (1992)), and the authors of this book prohibit publication of code which even partly contains routines from their book.

For this reason the computer programs were never made public and the only way we can make them public now seems to remove all Numerical Recipes routines and use equivalent (preferably better) routines from other sources or just to cook up these routines ourselves. This is indeed what we intend to do, and a first attempt is the treatment of the MLE, SMLE and hazard for the Competing Risk model under Current Status censoring, which has received a lot of attention from researchers, and is discussed at different spots in the book. We also give several approaches to the computation of confidence intervals for the current status model.

We try to link the `C/C++` programs to `R`, using `Rcpp`. In this way the programs are immediately available for Mac, Windows and Linux/Unix platforms and we also can connect to the graphical tools available in `R`. Moreover, using `C/C++` routines, the programs run very much faster than if we only would have implemented the routines in `R`. Parallel to this development, we also create GUI (Graphical User Interface) applications for Mac OS 10.10 (Yosemite), built in Xcode, allowing the user to open input files via the standard open file dialogue and run bootstrap simulations for computing confidence intervals.

For using `R`, using `Rcpp`, Windows users may have to install `Rtools`:
`http://cran.r-project.org/bin/windows/Rtools/index.html`
Mac users may have to use Xcode, or at least the corresponding command line tools, see (with advices for Windows, Mac and Linux users):
`https://support.rstudio.com/hc/en-us/articles/200486498-Package-Development-Prerequisites`
Mac users who use the GUI application, given in `compriskGUI.tar.gz` on:
`http://dutiosc.twi.tudelft.nl/~pietg/software.html`
do not have to download extra tools.

## 2. The competing risk model under current status censoring

### *2.1. The model*

The competing risk model under current status censoring is described on p. 121 of Groeneboom and Jongbloed (2014) and we take the formulation from there. Consider a situation where for a certain object there are several (say $K \in \mathbb{N}$) possible causes of failure and that at a random point $T$ in time the object is inspected. It is observed whether or not this object broke down before time $T$ or not (its ‘current status’). In case the object has broken down, it is also observed which of the $K$ possible causes (competing risks) lead to the breakdown. Write $X$ for the time of breakdown and $Y \in \{1,2,\dots,K\}$ for the corresponding cause. Together with inspection time $T$, the indicator vector

$$\Delta = (\Delta_1,\dots,\Delta_K) \text{ with } \Delta_k = 1_{[X\le T, Y=k]}$$

is observed. Note that if all indicators are zero, this means that $X > T$. If $X \le T$ also the breakdown cause is observed, so then exactly one of the $K$ indicators equals 1. Assuming $(X,Y)$ to have joint distribution function $F$ and $T$ to be independent of $(X,Y)$, this model is known as the competing risk model with current status observations. Data of this type arise naturally in cross-sectional studies with several failure causes.

Note that, given $T$, the vector

$$(\Delta_1,\Delta_2,\cdots,\Delta_K,\Delta_{K+1}) \text{ with } \Delta_{K+1} = 1-\sum_{k=1}^K \Delta_k \tag{2.1}$$

has a multinomial distribution with parameters 1 and $(F_1(T),\cdots,F_K(T),1-F_+(T))$, where

$$F_k(t) = P(X\le t, Y=k),\ \ t\ge 0, k=1,2,\dots,K,$$

is the sub-distribution function of $X$ for risk level $k$ and $F_+ = \sum_k F_k$ is the marginal distribution function of $X$. Denoting by $e_k$ the $k$th unit vector in $\mathbb{R}^K$, by $\#$ counting measure on $D = \{e_k : k=1,\dots,K+1\}$ and $G$ the distribution of $T$, we can define the measure $\mu = G\times\#$ on $\mathbb{R}\times D$. With respect to this (dominating) measure, the density of a single observation $(T,\Delta)$ is given by

$$p_F(t,\delta) = \prod_{k=1}^K F_k(t)^{\delta_k}(1-F_+(t))^{1-\delta_+}, \tag{2.2}$$

where $\delta_+ = \sum_{k=1}^K \delta_k$.

Now consider an independent sample of size $n$, distributed as $(T,\Delta_1,\dots,\Delta_K)$,

$$\left(T_i,\Delta^i\right) = \left(T_i,\Delta_1^i,\dots,\Delta_K^i\right),\ i=1,\dots,n,$$

where, for $1\le i\le n$

$$\Delta^i = \left(\Delta_1^i,\dots,\Delta_K^i\right) \text{ with } \Delta_k^i = 1_{\{X_i\le T_i,\,Y=k\}},\ k=1,\dots,K.$$

Also define

$$\Delta_{K+1}^i = 1-\sum_{k=1}^K \Delta_k^i = 1_{\{X_i>T_i\}}\,.$$

Using (2.2) and independence of the observations, the log likelihood (divided by $n$) is given by

$$\begin{aligned}\ell(F) &= \int \log p_F(t,\delta)\,d\mathbb{P}_n(t,\delta)\\ &= \int\Bigl\{\sum_{k=1}^K \delta_k\log F_k(t) + (1-\delta_+)\log(1-F_+(t))\Bigr\}\,d\mathbb{P}_n(t,\delta).\end{aligned} \tag{2.3}$$

where $\mathbb{P}_n$ is the empirical distribution of $(T_i,\Delta^i)$, $i=1,\dots,n$. *An MLE* $\hat F = (\hat F_1,\dots,\hat F_K)$ can then be defined by the property

$$\ell(\hat F) = \max_{F\in\mathcal{F}_K}\ell(F) \tag{2.4}$$

where

$$\begin{aligned}\mathcal{F}_K &= \{F=(F_1,\dots,F_K) : F_1,\dots,F_K \text{ are sub-distribution functions,}\\ &\qquad \text{such that for all } x\ge 0 : \sum_{k=1}^K F_k(x)\le 1\}.\end{aligned} \tag{2.5}$$

### *2.2. The Bangkok Metropolitan Administration cohort study*

The Bangkok Metropolitan Administration injecting drug users cohort study (Kitayaporn et al. (1998) and Vanichseni et al. (2001)) was started in 1995 to assess (among other things) the feasibility of conducting a phase III HIV vaccine efficacy trial for injecting drug users in Bangkok. The data on a subset of 1365 injecting drug users who were below 35 years of age in this study were analyzed by Maathuis and Hudgens (2011) and Li and Fine (2013). In this group, 392 were HIV positive, with 114 infected with subtype B, 237 infected with subtype E, 5 infected by another mixed subtype and 36 infected with missing subtype. The subjects with other, mixed or missing subtypes were grouped in a single category.

In Maathuis and Hudgens (2011), the maximum likelihood estimator (MLE) for these data is computed and also a so-called naive estimator, based on analyzing one category such as the type B subjects, ignoring the data on the other types. In Li and Fine (2013) both the regular MLE and a smoothed version of the MLE (called the SMLE) are computed and theory developed in Groeneboom, Jongbloed and Witte (2010) is used for constructing confidence intervals. They also estimate the hazard and construct confidence intervals for the hazard, again using Groeneboom, Jongbloed and Witte (2010). The regular MLE cannot directly be used for this purpose because it corresponds to a discrete distribution, so that some kind of smoothing is needed to estimate the hazard and to construct the confidence intervals.

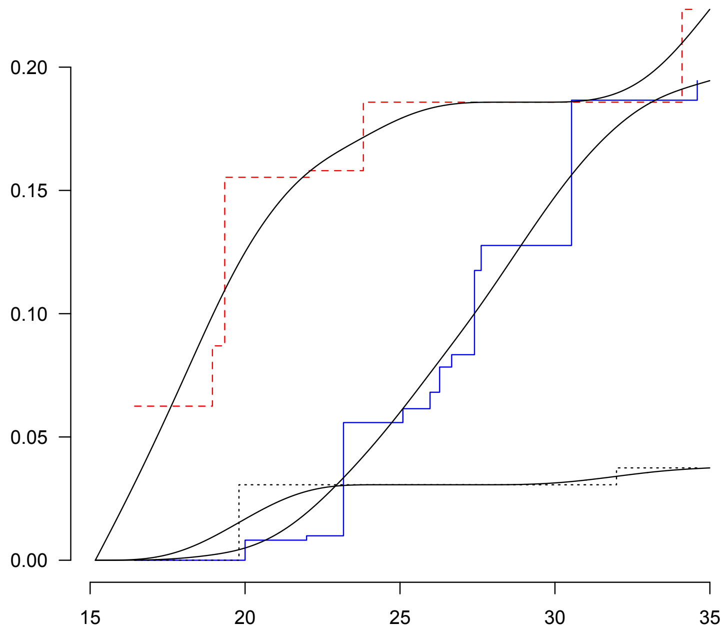


Fig 1: The MLE for the three categories in the Bangkok cohort study. The piecewise constant curves give the subdistribution functions, based on the MLE, for the different categories; dotted: type E, solid: type B, dashed; other types. The smooth solid curves give the corresponding estimates, based on the SMLE.

### *2.3. The iterative convex minorant algorithm*

The iterative convex minorant algorithm is derived from Corollary 2.10 in Groeneboom, Maathuis and Wellner, 2008, which is given below.

**Lemma 2.1** (Corollary 2.10, Groeneboom, Maathuis and Wellner, 2008). *Let $\lambda$ be given by*

$$\lambda = 1 - \int \frac{\delta_{K+1}}{1-\hat F_+(u)}\,d\mathbb{P}_n(u,\delta) \ge 0. \tag{2.6}$$

*Then $\hat F = (\hat F_1,\dots,\hat F_K)$ is an MLE if, for all $k=1,\dots,K$ and each point of jump $\tau_{ki}$ of $\hat F_k$:*

$$\int_{u\in[\tau_{ki},s)} \left\{\frac{\delta_k}{\hat F_k(u)} - \frac{\delta_{K+1}}{1-\hat F_+(u)}\right\} d\mathbb{P}_n(u,\delta) \ge \lambda 1_{[\tau_{ki},s)}(T_{(p)}),\ s\in\mathbb{R}, \tag{2.7}$$

*where equality holds if $s>\tau_{ki}$ is a point of increase of $\hat F_k$ or if $s>T_{(p)}$, where $T_{(p)}$ is the largest of the strictly ordered order statistics.*

To force (2.7) to hold, we can set up an iterative convex minorant algorithm. This algorithm is implemented in the `C++` file `icm.cpp` and in particular in the routine `ICM_iteration`. A so-called cusum diagram is formed, with points $(0,0)$ and points

$$\left(\sum_{i=1}^j w_{ki},\ \sum_{i=1}^j (w_{ki}y_{ki}+v_{ki}) - \lambda 1_{\{j=n_k\}}\right),\ j=1,\dots,n_k, \tag{2.8}$$

where $\lambda$ is as in (2.6), with $\hat F_+$ replaced by the $F_+$ at the present iteration step,

$$w_{ki} = \int_{u\in[T_i,T_j)} \left\{\frac{\delta_k}{F_k(u)^2} + \frac{\delta_{K+1}}{\{1-F_+(u)\}^2}\right\} d\mathbb{P}_n(u,\delta),$$

and $T_i$ and $T_j$ are successive points where $\delta_k = 1$,

$$v_{ki} = \int_{u\in[T_i,T_j)} \left\{\frac{\delta_k}{F_k(u)} - \frac{\delta_{K+1}}{1-F_+(u)}\right\} d\mathbb{P}_n(u,\delta),$$

and where $y_{ki}$ is the value of the subdistribution function $F_k$ at the point $T_i$ at the present iteration; $n_k$ is the number of points where $\delta_k = 1$. As explained in Groeneboom and Jongbloed (2014), the $w_{ik}$ have an interpretation via the diagonal of the Hessian matrix of the maximization problem.

We next determine the greatest convex minorant of the cusum diagram (2.8) and use its left derivative for forming the new values of $F_k$. The Lagrange multiplier $\lambda$ is computed after each iteration via (2.6), using the values of the $F_k$ at that iteration. We only have to determine the values of $F_k$ at the points where $\delta_k = 1$, since these are the only points where the subdistribution function can have mass. It can be seen that at a stationary point of these iterations the conditions (2.7) are satisfied and hence an MLE is found. To go from one iteration to the next we determine the step size by a golden section search algorithm.

In our experience, this algorithm is at present the fastest algorithm for finding the MLE. It can easily be extended to interval censoring with more observations, like case 2 interval censoring, or so-called mixed case interval censoring, but we concentrate for the moment on the current status model.

### *2.4. The SMLE*

The SMLE (smoothed maximum likelihood estimator) is computed using (9.75) in Groeneboom and Jongbloed (2014), where we use boundary correction on the left and right. The SMLE is defined on p. 367 of Groeneboom and Jongbloed (2014) for the data of the Bangkok cohort study, and given by

$$\tilde F_{nk,h}(t) = \int \left\{ I\!K\left(\frac{t-x}{h}\right) + I\!K\left(\frac{t+x-2a}{h}\right) - I\!K\left(\frac{2b-t-x}{h}\right)\right\} d\hat F_{nk}(x), \tag{2.9}$$

where $a=15$, $b=35$, $h$ is the bandwidth (taken to be $(b-a)n^{-1/5}$) and

$$\mathbb{K}(u) = \int_{-\infty}^u K(y)\,dy, \tag{2.10}$$

choosing for $K$ for the triweight kernel, given by

$$K(u) = \frac{35}{32}\left(1-u^2\right)^3 1_{[-1,1]}(u).$$

The SMLE is computed in the last routine `bdf` of `icm.cpp`.

As explained in Groeneboom and Jongbloed (2014), the SMLE has a faster asymptotic convergence rate ($n^{-2/5}$, the usual convergence rate of density estimation), if one is willing to assume smoothness, than the MLE (which has convergence rate $n^{-1/3}$ under the usual conditions). Also, the SMLE has asymptotically a normal limit distribution (under the appropriate conditions, given in Groeneboom and Jongbloed (2014)), in contrast with the MLE, for which the asymptotical distribution still has no analytic characterization.

## *2.5.* `Rcpp`

The `Rcpp` part of the algorithm is implemented in the routine `ComputeMLE(DataFrame input)` in `icm.cpp`. The input to the routine is a data frame which comes from a file with two columns: the first column contains the observation times and the second the causes of failure $1,\dots,K$ or 0 is there is no failure at that observation time. The R script `MLEThai.R` essentially only exists of the lines:

```
library(Rcpp)
icm<-sourceCpp("icm.cpp")
A<-read.table("dataThai.txt")
output <- ComputeMLE(A)
```

and has as output a list of three things: the MLE, the SMLE (computed on a equidistant grid of 1000 points) and the value of the log likelihood. The first line activates the `Rcpp` library (one may have to load the package `Rcpp` first), the second line compiles the `C++` file `icm.cpp`, the third line transforms the data into a form suitable for input to the function `ComputeMLE`, and the fourth line computes the MLE and SMLE from this function (and also produces the value of the log likelihood). The present data file `dataThai.txt` can of course be replaced by any data file of the same structure; the file should not have headers, and the number of observations and number of risks, which should have the labels $1,\dots,K$, is counted by the `C++` program `icm.cpp` itself.

The program also performs a reduction to take ties into account. For example, the data file `dataThai.txt` contains 1365 observations for three risks. After reduction for ties, the number of remaining (unique) observations is 1211, and at each unique observation the frequencies of the occurrences of $\delta_k = 1$ are given, for $k = 1,\dots,K$, where $K = 3$ in this case.

One can also use an input file where the ties are already taken into account, and where the data consist of the frequencies of the observations for the different risks are given. In that one case the input file contains $K + 2$ columns: the first column contains the observation times, the next column the frequencies for the observations where no failure is observed and next one gets the columns with the frequencies for the observed failures of type $k$, $k = 1,\dots,K$. The script demonstrating this approach is given in `MLEThai2.R`, and the input file is `dataThai2.txt`. The `C++` file for this approach is given by `icm2.cpp`.

For illustration purposes the drawing of Figure 1.7 on p. 11 of Groeneboom and Jongbloed (2014), where the MLE and SMLE are shown, is added to the scripts. This figure is also shown above in Figure 1.

## *2.6. Comparison with* `MLEcens`

In first instance, the `R` package `MLEcens` (Maathuis (2013)) was developed by Marloes Maathuis for analyzing bivariate interval censored data. For this type of data, the non-unicity of the MLE is more of an issue than for the present case of the MLE for competing risk data under current status censoring. The method proceeds by first computing rectangles which are candidates for containing mass and next computing the masses of the MLE on these rectangles. The really time consuming step of the algorithm is the computation of the MLE, not the preliminary reduction step of the computation of the candidate rectangles.

This method can also be used for computing the MLE for the present model, in which case the rectangles are replaced by intervals. The output of `MLEcens` gives indeed the masses of the MLE on these intervals, together with a list of the intervals. The computation of the MLE is based on an old `C` program, using support reduction, as discussed in Groeneboom, Jongbloed and Wellner (2008).

This algorithm is totally different from the iterative convex minorant algorithm. The support reduction algorithm successively builds up a distribution of mass from a few rectangles until further addition of new rectangles with mass will not increase the likelihood any more. This is done by two types of iteration: *inner iterations* which preform a least squares minimization, using weights given by the *outer iterations* which perform a change of the masses using Armijo's rule for determining a step size in a direction given by the inner iterations. Details of this procedure are given in Groeneboom, Jongbloed and Wellner (2008) for the so-called "Aspect experiment" in quantum statistics.

We compare the output of the two algorithms for the Bangkok data below. The iterative convex minorant algorithm gives on my computer, after 20 iterations, the following output for the MLE:

```
          [,1]        [,2]       [,3]       [,4]       [,5]
[1,]  16.43532 0.000000000 0.06250000 0.00000000 0.06250000
[2,]  18.94593 0.000000000 0.08695652 0.00000000 0.08695652
[3,]  19.34292 0.000000000 0.15531609 0.00000000 0.15531609
[4,]  19.80287 0.000000000 0.15531609 0.03060505 0.18592114
[5,]  20.00000 0.008140789 0.15531609 0.03060505 0.19406192
[6,]  21.98494 0.009897753 0.15531609 0.03060505 0.19581889
[7,]  22.06708 0.009897753 0.15801138 0.03060505 0.19851418
[8,]  23.17591 0.055833685 0.15801138 0.03060505 0.24445011
[9,]  23.81656 0.055833685 0.18579624 0.03060505 0.27223497
[10,] 25.09240 0.061458723 0.18579624 0.03060505 0.27786000
[11,] 25.96851 0.068139019 0.18579624 0.03060505 0.28454030
[12,] 26.27789 0.078359872 0.18579624 0.03060505 0.29476115
[13,] 26.66667 0.083361566 0.18579624 0.03060505 0.29976285
[14,] 27.40315 0.117539808 0.18579624 0.03060505 0.33394109
[15,] 27.61944 0.127664960 0.18579624 0.03060505 0.34406624
[16,] 30.53525 0.186595483 0.18579624 0.03060505 0.40299676
[17,] 31.98631 0.186595483 0.18579624 0.03745355 0.40984527
[18,] 34.09993 0.186595483 0.22342172 0.03745355 0.44747075
[19,] 34.59274 0.194506509 0.22342172 0.03745355 0.45538178
```

The first column gives observations (ages), the second to fourth column the values of the three subdistribution functions of which (at least) one has a jump at the corresponding observation time, and the 5th column gives the values of the sum function $F_+$ at these points. There are of course many more observations, but the MLE does not have jumps at these points.

After 20 iterations, the values of the estimates satisfy the two (Fenchel duality) conditions (7.56) and (7.57) of Groeneboom and Jongbloed (2014) at the level $10^{-10}$ (absolute value of inner product of nabla vector and solution smaller than $10^{-10}$ and minimum of partial sums of the nabla vector bigger than $10^{-10}$, respectively). The same convergence criterion is used in `MLEcens`.

If I apply `MLEcens` to the same data, I get the following output on my computer. For the masses on the rectangles I get the vector of values (denoted by $p in the output):

```
0.062500000 0.024456522 0.068359569 0.030605045 0.008140789 0.001756965
0.002695293 0.045935932 0.027784852 0.005625038 0.006680296 0.010220853
0.005001694 0.034178242 0.010125152 0.058930524 0.006848503 0.037625482
0.007911025 0.544618225
```

If we take the cumulative sums of these numbers, using the `R` function `cumsum`, we get:

```
0.06250000 0.08695652 0.15531609 0.18592114 0.19406192 0.19581889
0.19851418 0.24445011 0.27223497 0.27786000 0.28454030 0.29476115
0.29976285 0.33394109 0.34406624 0.40299676 0.40984527 0.44747075
0.45538178 1.00000000
```

in which we recognize the last column of the output of the iterative convex minorant algorithm, corresponding to the sum function $F_+$, except for the last number (which just gives the jump to the total mass).

As next output, I get the table $rects in the output:

```
          [,1]     [,2]     [,3] [,4]
[1,]  15.16769 16.43532 1.75 2.25
[2,]  18.94045 18.94593 1.75 2.25
[3,]  19.34018 19.34292 1.75 2.25
[4,]  19.79192 19.80287 2.75 3.25
[5,]  19.99179 20.00000 0.75 1.25
[6,]  21.97399 21.98494 0.75 1.25
[7,]  22.05065 22.06708 1.75 2.25
[8,]  23.15400 23.17591 0.75 1.25
[9,]  23.80287 23.81656 1.75 2.25
[10,] 25.04860 25.09240 0.75 1.25
[11,] 25.94114 25.96851 0.75 1.25
[12,] 26.26968 26.27789 0.75 1.25
[13,] 26.64750 26.66667 0.75 1.25
[14,] 27.40041 27.40315 0.75 1.25
[15,] 27.61670 27.61944 0.75 1.25
[16,] 30.52977 30.53525 0.75 1.25
[17,] 31.97262 31.98631 2.75 3.25
[18,] 34.09719 34.09993 1.75 2.25
[19,] 34.54894 34.59274 0.75 1.25
[20,] 34.99521 100.0000 0.75 3.25
```

Here one recognizes in the second column the numbers in the first column of the table, produced by the iterative convex minorant. The numbers 1.75, 2.25, etc. in the last columns may seem somewhat peculiar, but have the following meaning. The interval $[0.75, 1.25]$ corresponds to risk 1, the interval $[1.75, 2.25]$ to risk 2, the interval $[2.75, 3.25]$ to risk 3, and the interval $[0.75, 3.25]$ to an observation where there is no failure. This way of coding shows the descendance of the algorithm from the algorithm for bivariate interval censoring.

So, if one places the mass of the intervals at the right endpoint, one gets the same results as with the iterative convex minorant algorithm. It is also seen that the numbers in the first two columns are pretty close, except for the irrelevant last row (as an aside, one could also argue that this last interval should start at 34.59274 rather than at 34.99521), so there is only a slight space of freedom for defining different MLEs.

To further compare the two algorithms for a more challenging data set, we compared the performance for the example of 25000 observations on data, generated by the exponential-type subdistribution functions $F_k(t) = (k/3)\{1 - e^{-kt}\}$, $t \geq 0$, $k = 1, 2$, and analyzed on pages 190 and 191 of Groeneboom and Jongbloed (2014). Using the script `MLE25000.R`, the computation for and plotting of Figure 2 below, where the SMLE also was computed, took about 10 seconds on my computer. On the other hand, just to compute the MLE took 100 seconds using the package `MLEcens`. If the number of observations is growing, the algorithm in `MLEcens` is considerably slowed down by the matrix inversions (or growing number of linear equations to be solved), whereas the iterative convex minorant algorithm does not have to solve equations of this type (only using the diagonal of the Hessian matrix).

## 3. Bootstrap confidence intervals for the hazards in the competing risk model

Confidence intervals for the hazards in Bangkok data, discussed above, were given in Li and Fine (2013). The intervals, given there, are based on asymptotic theory and do not use the bootstrap. It is not at all trivial that the bootstrap can be used for the confidence intervals, if one wants to base this on the computation of the MLE, since the bootstrap for the MLE itself will no doubt fail, as one can infer from Kosorok (2008) and Sen, Banerjee and Woodroofe (2010). The reason that one still can get the confidence interval by bootstrapping seems to be a consequence of the fact that the estimate of the hazard, based on the MLE, has an asymptotic behavior described by (what is called in Groeneboom and Jongbloed (2014)) *local smooth functional theory*,

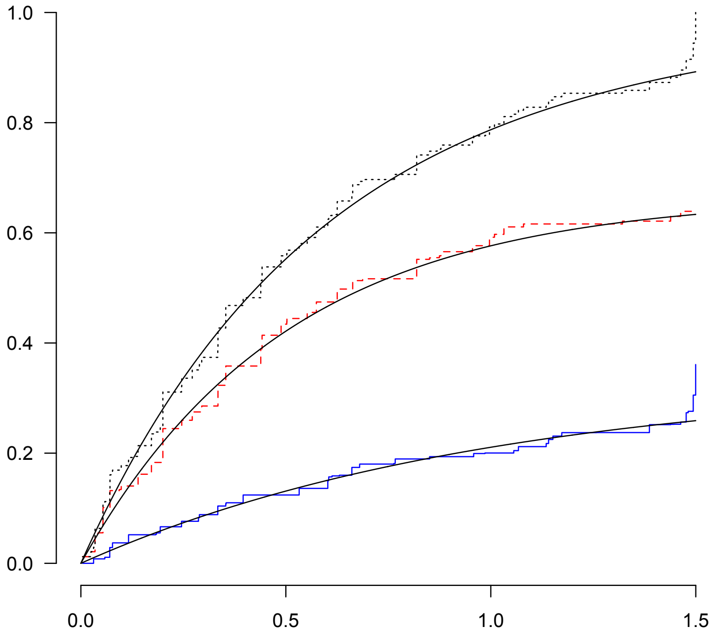


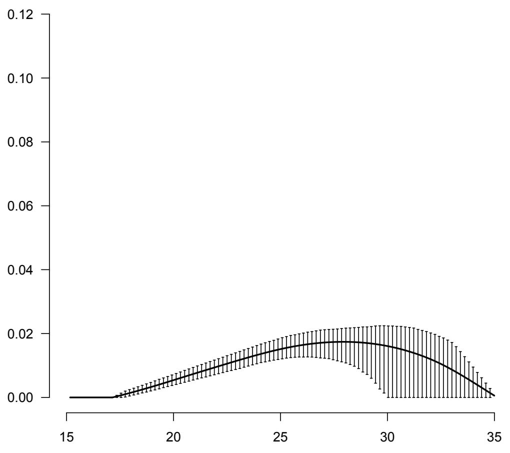
Fig 2: The two distribution functions $F_k(t) = (k/3)\{1 - e^{-kt}\}$ for the competing risk model with current status data and their MLE's for a sample of size $n = 25000$. The upper curves give the sum function $F_+$ and its estimator $\hat{F}_{n+}$.

which "washes away" the peculiar local limit behavior of the MLE. Nevertheless, the theory for this is far from complete and is described by integral equations one has to be able to solve first, as also argued in Groeneboom (2013). And once one has solved the integral equations, one has to deduce the relevant equivalence to a "toy estimator" from this, which usually is also a somewhat daunting task.

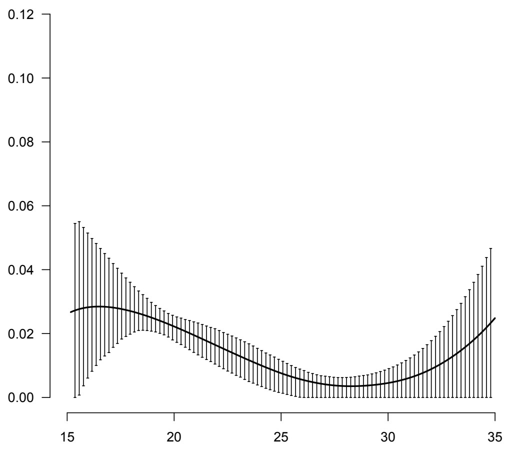
We will not further dwell on these matters here, but instead describe the `R` script. We take the data set `dataThai.txt` again and activate the `C++` program via the `R` script `MLEThai.R`. This will produce three hazard estimates for the three groups and also 95% confidence intervals for the hazards, computed at 99 points of the interval $[15, 35]$. Two of the pictures of the confidence intervals can be seen in Figure 12.2 on p. 369 of Groeneboom and Jongbloed (2014). We give the the three pictures below. In Groeneboom and Jongbloed (2014) both studentized an non-studentized confidence intervals are discussed; we give the results for the non-studentized ones, since there was not much difference anyway. The `C++` code `icm.cpp` can be consulted to see what is done: 1000 bootstrap samples are drawn (with replacement) form the original data, consisting of the pairs $(T_i, \Delta_i)$, and for each sample the MLE is computed iteratively, using the iterative convex minorant algorithm. Once the MLE is computed, one can compute the SMLE and the density estimate, and on the basis of these the hazard estimate is computed for the bootstrap sample. It is interesting to note that they look somewhat different from the corresponding intervals for the same data in Li and Fine (2013), which might illustrate the distance between asymptotic theory and bootstrap approximation.

## 4. A GUI application and confidence intervals for the SMLE

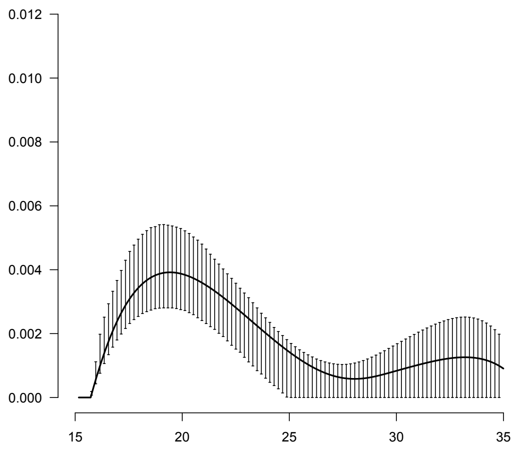
As mentioned in the Introduction, we also started writing GUI (Graphical User Interface) applications, parallel to the development of the `Rcpp` approach. These applications were developed for Mac OS X, using



(a)



(b)

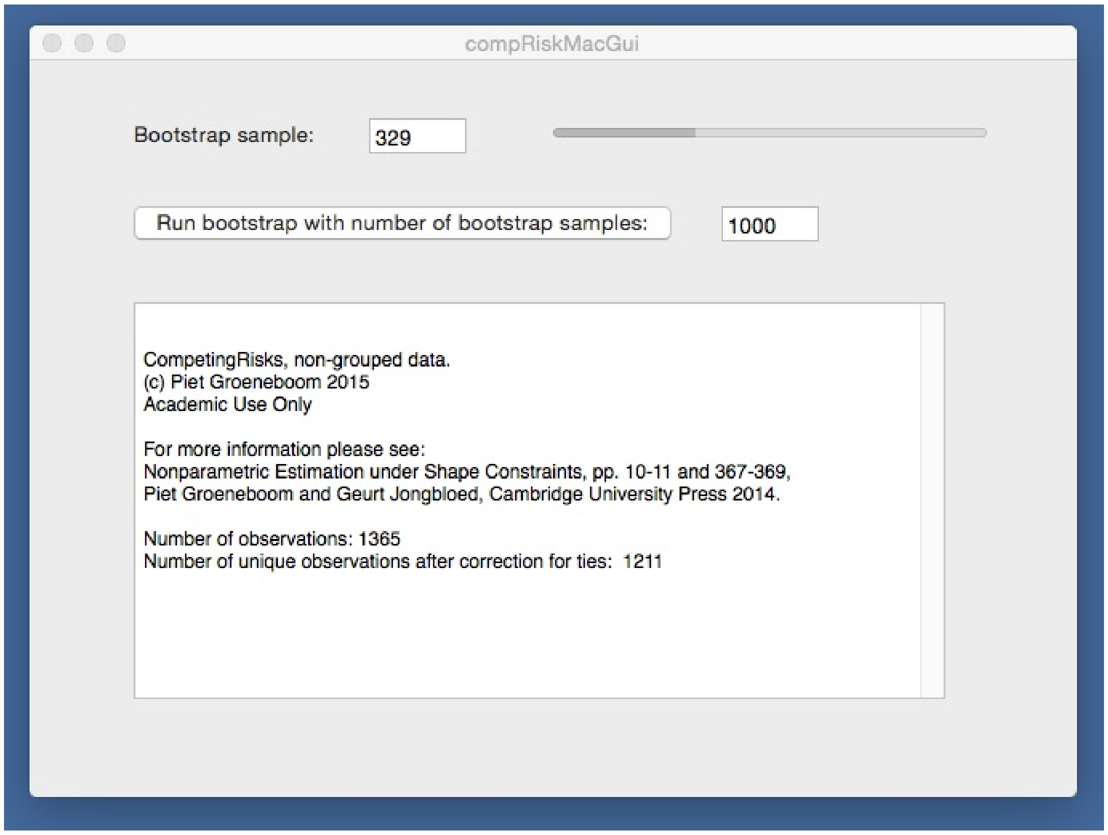


(c)

Fig 3: 95% confidence intervals for the hazards for the group infected with type B (Figure (a)), with type E (Figure (b)) and remaining group (Figure (c)) in the Bangkok cohort data.

Xcode and a combination of `Objective-C` and `C++`, but it should also be possible to implement these for Windows, using Microsoft Visual `C++`.

The application allows the user to specify the number of bootstrap samples to be taken for the computation of the confidence intervals. It has a menu, where one can choose the input file by the standard "File-open $\cdots$" submenu. One can also choose via the submenu "Statistic" between confidence intervals for the SMLE or confidence intervals for the hazard. One can both use the ungrouped and grouped file formats. In the latter case the observations are unique and at each such unique point the information about the frequencies of failures for the competing risks is given, and the frequency of observations of no failure at that point. The screenshot Figure 4 shows a moment in the analysis of the Bangkok data, where the number of observations was 1365, and where after the correction for ties 1211 unique observations remained. To have an idea of



```
CompetingRisks, non-grouped data.
(c) Piet Groeneboom 2015
Academic Use Only

For more information please see:
Nonparametric Estimation under Shape Constraints, pp. 10-11 and 367-369,
Piet Groeneboom and Geurt Jongbloed, Cambridge University Press 2014.

Number of observations: 1365
Number of unique observations after correction for ties:  1211
```

Fig 4: A screenshot of the GUI application

the progress of the bootstrap samples, there is a (blue) progress bar, showing the proportion of bootstrap samples that have been treated. The number of treated bootstrap samples is also shown in the (so-called) textfield left if it. See the discussion on Section 2.5 for the difference between the input file with two columns and 1365 observation points for the ungrouped data and the input file with $K + 2$ (in this case: 5) columns and 1211 (unique) observation points for the grouped data.

For performing the bootstrap, it seemed easiest to stick with the 1365 observation points, from which we resample uniformly 1365 bootstrap sample points of type $(T_i, \Delta^i)$ and compute the statistics for the bootstrap sample. This means that in the case of the input of the 1211 (grouped) data points, the data were first transformed to the other format, where we only have two columns with 1365 points $(T_i', \gamma_i)$, where $\gamma_i \in \{0, 1, \dots, K\}$ denotes the risk for which there is a failure at time $T_i'$ (or equals zero if it denotes an observation for which there is no failure).

Since the computations are again done by the `C++` part of the program, the computations take about the same time as in the `Rcpp` implementation. For example, the bootstrap experiment with 1000 bootstrap samples of 1365 observations for the confidence intervals for the hazards takes about 20 seconds (the amount of time is computed in the program and given in the printed output). One can also compute the 95% confidence intervals for the SMLE. This takes somewhat longer on my computer, about 50 seconds, since in this case the book on p. 367 is (again) followed, in taking a kind of “studentized" confidence intervals, for which an estimate of the variance has to be computed for each sample, see (12.7) on p. 367. This was done in view of the “reproducibility" of Figure 12.1 on p. 368. It turns out, however, that, just as in the case of the confidence intervals for the hazards, the non-studentized confidence intervals for the SMLE, not using this scaling by the estimate of the standard deviation, are rather similar.

For the computation of the confidence intervals for the MLE for current status and interval censored data, a different kind of bootstrap is proposed in Sen and Xu (2015). They suggest bootstrapping from the SMLE, by resampling only the indicators, with probabilities given by the SMLE, and using the (fixed) observation points of the original sample. For example, in the current status model the bootstrap samples contain the observations $(T_i, \Delta_i^*)$, where the $T_i$ are the observations of the original sample, and $\Delta_i^*$ is, conditionally on the original sample, a Bernoulli random variable, with success probability $\tilde F_{n,h}(T_i)$ and where $\tilde F_{n,h}$ is an SMLE, with a bandwidth $h$, tending to zero slower than $n^{-1/3}\log n$. They prove in their Theorem 3 in Section 2.3 (in the supplementary material to their paper) that this leads to consistent bootstrapping of the MLE. This could also be extended to confidence intervals for the SMLE in the competing risk model, by smoothing the bootstrap values of the MLE.

In contrast, we would, for the current status model, resample the $(T_i, \Delta_i)$ (so both the values $T_i$ and $\Delta_i$ become random in our bootstrap sample) and compute the MLE and subsequently the SMLE directly on the basis of such a bootstrap sample. It is indeed surprising that this leads to inconsistent estimates of the distribution of the MLE, but consistent estimates of the corresponding distribution of the SMLE. The latter seems to be a consequence of the fact that the SMLE, although based on the MLE, seems to “by-pass" the aberrant limit behavior of the bootstrap version of the MLE.

The bootstrap confidence intervals of our method for the Thailand data are shown in Figure 5.

## 5. Confidence intervals for the current status model (see Section 9.5 of Groeneboom and Jongbloed (2014) and Groeneboom and Jongbloed (2015))

The current status model is the simplest interval censoring model, where the sample is of the form

$$(T_i, \Delta_i),\ i = 1, \dots, n,$$

and $\Delta_i = 1_{\{X_i \le T_i\}}$, where $X_i$ and $T_i$ are independent real-valued random variables. The model is also called the “interval censoring, case 1” model. Since the current status model is the (simplest) prototype of a number of inverse statistical models, we will give a number of examples of it on the software site and will in particular study the confidence intervals and the bootstrap models for this model.

The MLE $\hat F_n$ can be very simply computed by computing the left derivative of the greatest convex minorant of the cusum diagram, formed by the point $(0, 0)$ and the points

$$\left(i, \sum_{j=1}^{i} \Delta_j\right),\ i = 1, \dots, n,$$

and the SMLE is of the form

$$\tilde F_{n,h}(t) = \int \left\{ I\!\!K\left(\frac{t-x}{h}\right) + I\!\!K\left(\frac{t+x-2a}{h}\right) - I\!\!K\left(\frac{2b-t-x}{h}\right)\right\} d\hat F_n(x), \tag{5.1}$$

where $h$ is the chosen bandwidth and $I\!\!K$ is the integrated kernel, defined by (2.10) (see also (2.9)), if the observations are concentrated on the interval $[a, b]$.

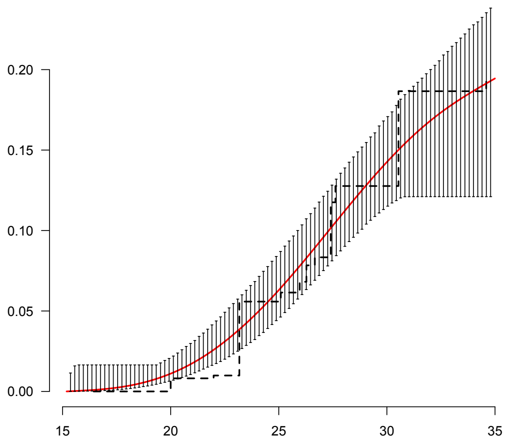
### *5.1. Xcode and Microsoft Visual C++ programs for simulations of confidence intervals, based on the SMLE, using the naive bootstrap.*

We start by considering a simulation study for the model where the $X_i$ are generated by the truncated exponential distribution with density

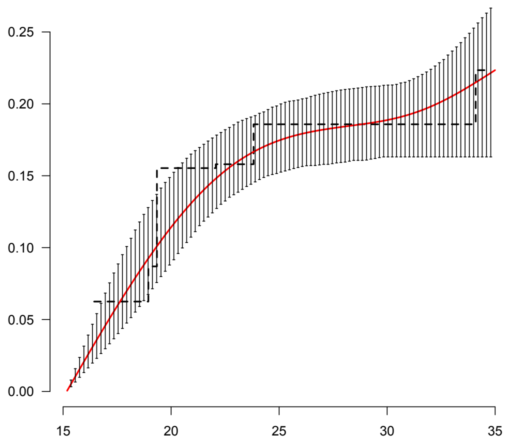
$$f(x) = \frac{e^{-x}}{1-e^{-2}},\ x \in [0, 2],$$

and where the observation times $T_i$ are generated by the uniform density on $[0, 2]$. Here we can compute confidence intervals for the SMLE in a similar way as we did in the computing risk model, and a typical picture of what we get in this case is given in Figure 6. The confidence intervals are constructed using the scaling by an estimate of the variance at the point $t$ (neglecting the factor $g(t)^2$ in the denominator):

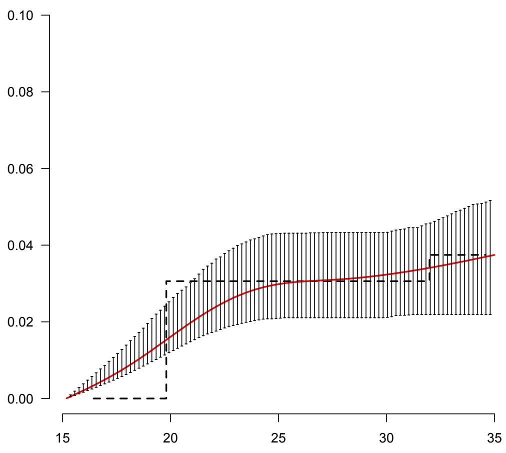
$$S_n(t)^2 = n^{-2} \sum_{i=1}^{n} \{K_h(t - T_i) - K_h(t + T_i - 2a) - K_h(2b - t - T_i)\}^2 \left(\Delta_i - \hat F_n(T_i)\right)^2, \tag{5.2}$$



(a)



(b)



(c)

Fig 5: 95% confidence intervals for the subdistribution functions for the group infected with type B (Figure (a)), with type E (Figure (b)) and remaining group (Figure (c)) in the Bangkok cohort data.

with the corresponding estimate for the bootstrap samples, see (9.76) in Groeneboom and Jongbloed (2014).

Here $K_h(u) = h^{-1}K(u/h)$ and $K$ is the usual symmetric kernel for density estimation with support $[-1,1]$, for example the Triweight kernel (which is the one we use). Note that, although the SMLE is defined by the *integrated* kernel $\mathbb{K}$, the variance is defined by using its derivative $K$. The estimate follows from an integration by parts argument from the definition of the SMLE (5.1). The same procedure was followed in the competing risk model. Simulation experiments for this model are reported in part (C) of Section 9.5 of Groeneboom and Jongbloed (2014) and can be reproduced by using the terminal application `curstat_bootstrap_simulation` for Mac or `curstat.exe` for Windows, which are contained in the directories `curstat_bootstrap_simulations_Mac` and `curstat_bootstrap_simulations_Windows`, respectively, or by compiling the Xcode or Microsoft Visual C++ 2013 project, also contained in these directories, and

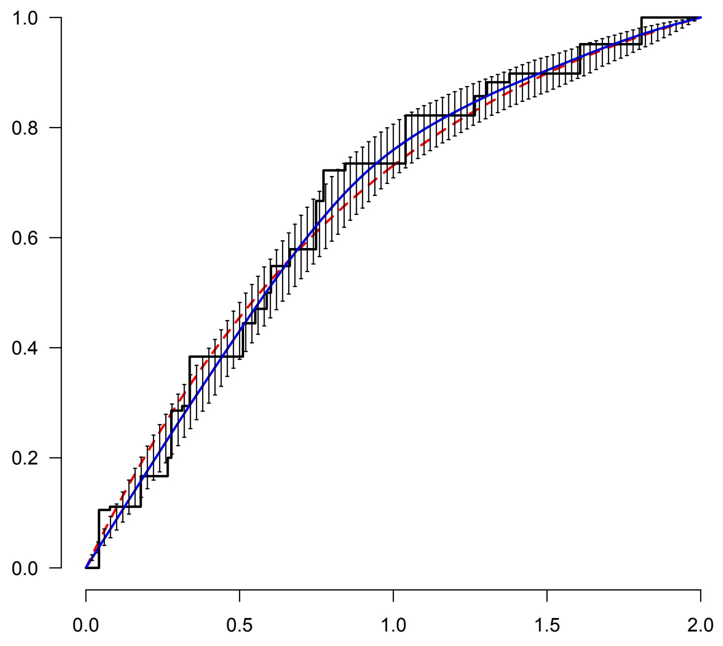


Fig 6: 95% confidence intervals for the distribution function $F_0(x) = \{1-e^{-x}\}/\{1-e^{-2}\}$, $x \in [0,2]$, based on 1000 bootstrap samples drawn with replacement from a sample $(T_1,\Delta_1),\dots,(T_n,\Delta_n)$ in the current model discussed in the text, where $n = 1000$. The (red) dashed curve is the real underlying truncated exponential distribution function and the smooth blue curve is the SMLE, based on the original sample. The piecewise constant curve is the MLE in the original sample.

running the application from the project. The simulations for 1000 samples, with 1000 bootstrap samples from each sample, take about 2 hours on my Mac and about 5 hours on my Windows laptop, although the `C++` code is the same. For generating the simulation samples, the Mersenne twister is used. For this we use the input file `<random>`, which is part of `C++11`, but which unfortunately still cannot be used in `Rcpp`.

For each generated original sample in the outer iterations (the 1000 bootstrap samples constituting the inner iteration loop), the number of the iteration, the value of the real underlying distribution function at $t = 1$, the SMLE at $t = 1$, the lower 2.5% and upper 97.5% values using the corresponding 1000 bootstrap samples, and the growing number of non-covered values of $F_0(1)$ is printed in the terminal window (in this order).

Analogously to the competing risk model, an alternative is to generate bootstrap samples from the SMLE $\tilde F_{nh}$, by leaving the observation times fixed and generating the indicators $\Delta_i^*$ as Bernoulli random variables, with succes probability $\tilde F_{nh}(T_i)$, as is done in Sen and Xu (2015).

### *5.2. R script for simulations of confidence intervals, based on the SMLE, using the naive bootstrap and bias correction*

We took the 99 equidistant points $0.02, 0.04, \dots, 1.98$ in the interval $(0,2)$, and computed the 95% confidence intervals at each of these points via

$$\left[\tilde F_{nh}(t) - \beta_h(t) - U^*_{1-\alpha/2}(t)S_n(t), \tilde F_{nh}(t) - \beta_h(t) - U^*_{\alpha/2}(t)S_n(t)\right], \tag{5.3}$$

where $S_n(t)$ is given by (5.2), see formula (9.78) in Groeneboom and Jongbloed (2014), and where the real bias is in second order:

$$\beta_h(t) = \begin{cases} -\dfrac{h^2e^{-t}\int u^2K(u)\,du}{2\{1-e^{-2}\}} & , t \in [h, 2-h], \\ -\dfrac{h^2e^{-t}\left\{\int u^2K(u)\,du - 2\int_v^1(u-v)^2K(u)\,du\right\}}{2\{1-e^{-2}\}}, \qquad v = \dfrac{t}{h} & , t \in [0,h), \\ -\dfrac{h^2e^{-t}\left\{\int u^2K(u)\,du - 2\int_v^1(u-v)^2K(u)\,du\right\}}{2\{1-e^{-2}\}}, \qquad v = \dfrac{2-t}{h} & , t \in (2-h, 2], \end{cases}$$

see pp. 271 and 272 of Groeneboom and Jongbloed (2014). Evaluating the integrals, we get:

$$\beta_h(t) = \begin{cases} -\dfrac{h^2e^{-t}}{18\{1-e^{-2}\}} & , t \in [h, 2-h], \\ -\dfrac{h^2e^{-t}\left\{35v/64 - v^2 + 35v^3/48 - 7v^5/32 + v^7/16 - 5v^9/576\right\}}{2\{1-e^{-2}\}}, \qquad v = \dfrac{t}{h} & , t \in [0,h), \\ -\dfrac{h^2e^{-t}\left\{35v/64 - v^2 + 35v^3/48 - 7v^5/32 + v^7/16 - 5v^9/576\right\}}{2\{1-e^{-2}\}}, \qquad v = \dfrac{2-t}{h} & , t \in (2-h, 2], \end{cases}$$

Moreover, $U_\alpha^*(t)$ is the $\alpha$th percentile of the 1000 values:

$$Z_{n,h}^*(t) = \frac{\tilde F_{nh}^*(t) - \tilde F_{nh}(t)}{\sqrt{n^{-2}\sum_{i=1}^n \left\{K_h(t-T_i^*) - K_h(t+T_i^*) - K_h(4-t-T_i^*)\right\}^2 \left(\Delta_i - \hat F_n^*(T_i^*)\right)^2}}, \tag{5.4}$$

where $\hat F_n^*$ is the ordinary MLE (not the SMLE!) of the bootstrap sample $(T_1^*, \Delta_1^*), \dots, (T_n^*, \Delta_n^*)$.

The bootstrap confidence interval is inspired by the (non-trivial) fact that the SMLE is asymptotically equivalent to the toy estimator ("toy" because we cannot use it in practice, since we know neither $F_0$ nor $g$):

$$F_{nh}^{toy}(t) = \int \left\{I\!K_h(t-u) + I\!K_h(t+u) - I\!K_h(4-t-u)\right\}\,dF_0(u) + \frac{1}{n}\sum_{i=1}^n \frac{\left\{K_h(t-T_i) - K_h(t+T_i) - K_h(4-t-T_i)\right\}\left\{\Delta_i - F_0(T_i)\right\}}{g(T_i)},$$

see (11.53) in Groeneboom and Jongbloed (2014), which has sample variance

$$S_n(t)^2 = \frac{1}{n^2}\sum_{i=1}^n \frac{\left\{K_h(t-T_i) - K_h(t+T_i) - K_h(4-t-T_i)\right\}^2\left\{\Delta_i - F_0(T_i)\right\}^2}{g(T_i)^2},$$

and also by Theorem 11.10 in Groeneboom and Jongbloed (2014), which tells us that, if $h \sim cn^{-1/5}$, under the conditions of that theorem, for each $t \in (0,2)$,

$$n^{2/5}\left\{\tilde F_{nh}(t) - F_0(t)\right\} \xrightarrow{\mathcal{D}} N\left(\mu, \sigma^2\right), \qquad n \to \infty,$$

where

$$\mu = \tfrac12 c^2 f_0'(t)\int u^2K(u)\,du$$

and

$$\sigma^2 = \frac{F_0(t)\{1-F_0(t)\}}{cg(t)}\int K(u)^2\,du,$$

Of course, in real life we do not have the expression for the real bias in second order, but it is of interest to see how the confidence intervals turn out to be, if we use the bootstrap procedure with this bias correction.

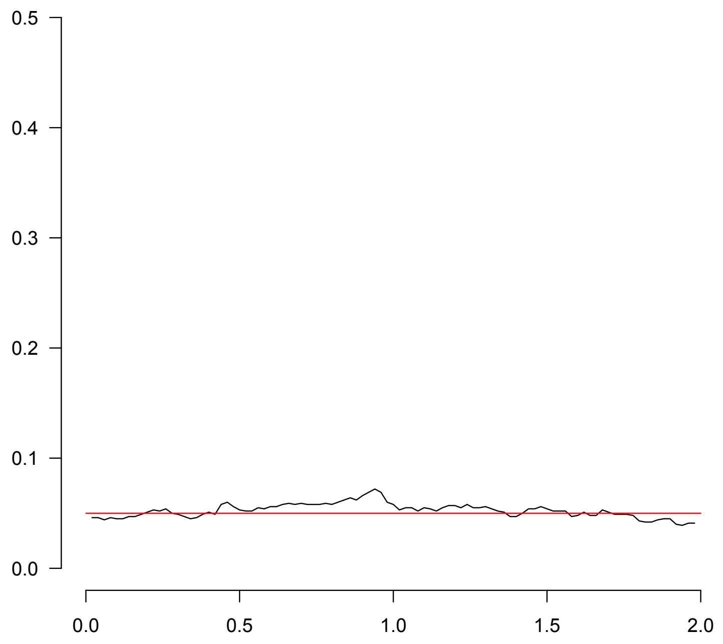


Fig 7: Truncated exponentials. Proportion of times that $F_0(t_i)$, $t_i = 0.02, 0.04, \dots$ is not in the CI's in 1000 samples $(T_1, \Delta_1) \dots, (T_n, \Delta_n)$, using the bias corrected confidence intervals and the SMLE bootstrap confidence intervals. The red line represents the function $y = 0.05$.

It turns out that if we use the exact second order expression for the bias, the confidence intervals give an excellent coverage, in fact, a better coverage than any of the other methods we tried.

A picture of the graph of misclassifications if given in Figure 7. This picture can be reproduced by using the Rcpp script `CI_SMLEcurstat.R`. The script can be found in the directory `CI_SMLE_simulation` and uses the `C++` file `curstat_bootstrapSMLE.cpp`. However, it takes about two hours on my laptop, so it is typically something to be done at night or in the background of other activities. The latter method can be used when the script is run in RStudio. Also shown are the outer iterations, in which the original sample is generated and the confidence interval is computed by taking 1000 bootstrap samples (the inner iterations). The upper and lower bounds of the confidence interval are shown and also a counter, counting the number of times $F_0(1)$ is not in the corresponding interval. At the end of the 1000 outer iterations one gets a number that usually is slightly above or below the intended number 50. One can also run the script directly in R, but in my experience with the GUI application on Mac OS X, the iterations are shown in a somewhat erratic ("chunky") way, and it is also more difficult to let it run in the background than it was using RStudio.

The real challenge here is to find a good *estimate* of the bias, a problem that still has not been solved. An alternative is to use undersmoothing, for example using the bandwidth $2n^{-1/4}$, see the discussion on page 272 of Groeneboom and Jongbloed (2014).

## 5.3. *R script for simulations of confidence intervals, based on the MLE, using bootstrapping from the SMLE*

It is argued in Sen and Xu (2015) that the bootstrap, where the MLE is recomputed in samples from the MLE in the orignal sample, will fail to produce consistent confidence intervals. Similarly, it follows by arguments analogous to the arguments in Kosorok (2008) that the classical bootstrap, where samples with replacement are drawn from the original sample $(T_1,\Delta_1),\dots,(T_n,\Delta_n)$ cannot be expected to be consistent.

However, Sen and Xu (2015) show that if $F_0$ and the observation distribution function $G$ are continuously differentiable at an interior point $t_0$, with derivatives $f_0(t_0)>0$ and $g(t_0)>0$, respectively, and if $F_n$ is an estimator such that

$$\lim_{n\to\infty}\sup_x |F_n(x)-F_0(x)|=0, \tag{5.5}$$

almost surely, and if, uniformly for $t$ in bounded intervals,

$$\lim_{n\to\infty}\left|F_n(t_0+n^{-1/3}t)-F_n(t_0)-n^{-1/3}f_0(t_0)t\right|=0, \tag{5.6}$$

almost surely, then, conditionally on the data, the distribution of $n^{1/3}\{\hat F_n^*(t_0)-F_0(t_0)\}$, where $\hat F_n^*$ is the MLE of $F_0$, based on a sample $(T_1,\Delta_1^*),\dots,(T_n,\Delta_n^*)$, and $\Delta_i^*$ is a Bernoulli variable with success probability $F_n(T_i)$, converges, almost surely, weakly to the same limit distribution as $n^{1/3}\{\hat F_n(t_0)-F_0(t_0)\}$, where $\hat F_n$ is the MLE in the original sample. In particular, we can take $F_n$ equal to the SMLE, and get consistent confidence intervals in this way. An attractive feature of this procedure is that we keep the observation times $T_i$ fixed and equal to the original observation times, and only resample the $\Delta_i^*$.

The SMLE $\tilde F_{nh}$ has the derivative

$$\tilde f_{nh}(t)=\int\{K_h(t-x)+K_h(t+x-2a)+K_h(2b-t-x)\}\,d\hat F_n(x), \tag{5.7}$$

see (5.1), which is similar to a kernel density estimate with the Schuster boundary correction at the left and right endpoints $a$ and $b$, respectively, where the MLE $\hat F_n$ replaces the empirical distribution function $\mathbb{F}_n$. It is clear that the SMLE will therefore satisfy (5.5) and (5.6), under the appropriate conditions for the bandwidth $h$ and the underlying distribution functions, further discussed in Sen and Xu (2015).

Considering the model with the truncated exponential on $[0,2]$ again, as in the preceding subsection, we take the SMLE $\tilde F_{nh}$, with $h=2n^{-1/5}$. We generate 1000 samples $(T_1,\Delta_1),\dots,(T_n,\Delta_n)$ of size $n=1000$ from the truncated exponential distribution $F_0$ on $[0,2]$ and the Uniform$(0,2)$ distribution as observation distribution. For each of these 1000 samples we compute the MLE $\hat F_n$ and SMLE $\tilde F_{nh}$, where the chosen bandwidth is $h=2n^{-1/5}$, and generate 1000 bootstrap samples $(T_1,\Delta_1^*),\dots,(T_n,\Delta_n^*)$, where $\Delta_i^*$ is a Bernoulli variable with success probability $\tilde F_{nh}(T_i)$. For the 1000 bootstrap samples we determine the 2.5th and 97.5th percentiles $U_{\alpha/2}^*(t)$ and $U_{1-\alpha/2}^*(t)$, $\alpha=0.05$, of the values

$$\hat F_n^*(t)-\tilde F_{nh}(t),\ t=0.02,0.04,\dots,1.98,$$

where $\hat F_n^*(t)$ is the MLE of the bootstrap sample. The 95% confidence interval for $F_0(t)$ is then given by:

$$\left[\hat F_n(t)-U_{0.975}^*(t),\hat F_n(t)-U_{0.025}^*(t)\right],$$

where $\hat F_n$ is the MLE of the original sample. A picture of the confidence intervals is shown in Figure 8. Note the pronounced non-monotonicity which is caused by the fact that the downward trend of the percentiles $U_\alpha^*(t)$ is not sufficiently compensated by the upward trend of the SMLE on flat pieces of the MLE $\hat F_n$.

Pictures of the proportion of non-coverages of the real value of the distribution in 1000 samples, comparable to Figure 7, are shown in Figure 9. The coverage is off at the boundary but rather well on target on the interval $[0.2,1.6]$.

The pictures can be reproduced by running the R script CI_Sen_Xu_curstat.R in the directory CI_Sen_Xu_simulation, which, in turn, belongs to the directory Confidence Intervals (current status simulations). It takes about three minutes on my laptop.

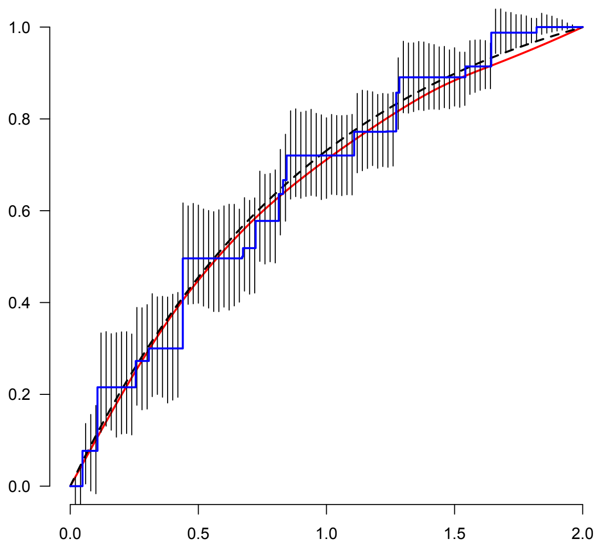


Fig 8: Sen-Xu confidence intervals for the (truncated exponential) distribution function $F_0$, based on a sample of 1000 observations $(T_i, \Delta_i)$, where $T_i$ is Uniform$(0, 2)$. The intervals are based on 1000 (smooth) bootstrap samples of indicators $\Delta_i^*$, using the SMLE (red curve) of the original sample. The dashed curve is the truncated exponential distribution function.

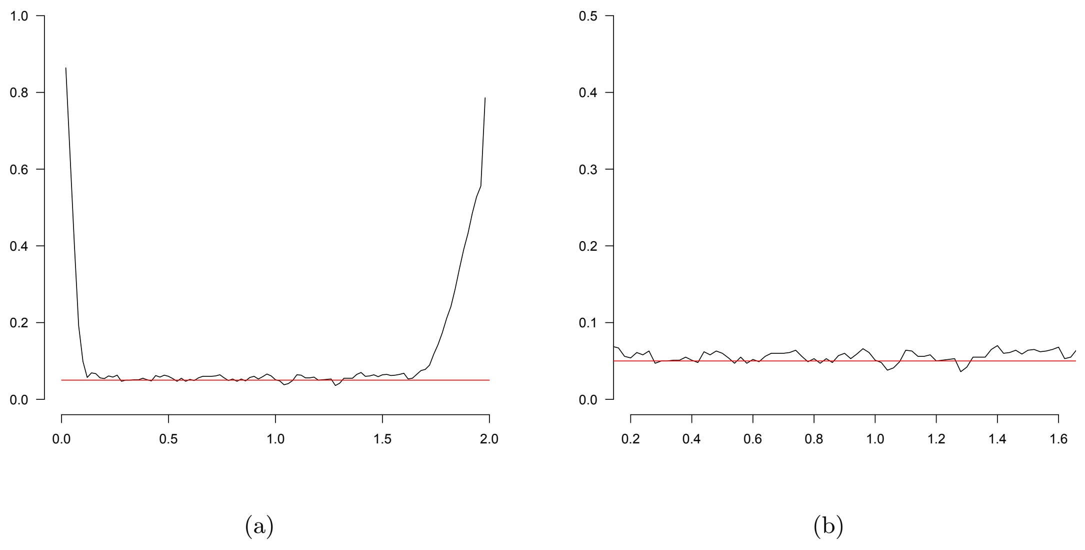


Fig 9: Sen-Xu confidence intervals. Proportion of times that $F_0(t_i)$, $t_i = 0.02, 0.04, \ldots$ is not in the CI's in 1000 samples $(T_1, \Delta_1) \ldots, (T_n, \Delta_n)$; (a): on the interval $[0, 2]$, (b): on the interval $[0.2, 1.6]$.

### 5.4. R script for simulations of confidence intervals, based on the LR test for the MLE

The LR test based intervals of Banerjee and Wellner (2001, 2005) are based on inverting the acceptance region of the LR test for the null hypothesis $F_0(t_0) = \theta$ and use the fact that, as $n \to \infty$,

$$2n\left\{\int\left\{\delta\log\hat F_n(t)+(1-\delta)\log\left(1-\hat F_n(t)\right)\right\}\,d\mathbb{G}_n(t)\right.$$
$$\left.-\int\left\{\delta\log\hat F_n^{(0)}(t)+(1-\delta)\log\left(1-\hat F_n^{(0)}(t)\right)\right\}\,d\mathbb{G}_n(t)\right\}\xrightarrow{\mathcal{D}}\mathbb{D},$$

where $\hat F_n$ and $\hat F_n^{(0)}$ are the nonrestricted MLE and the MLE restricted by the null hypothesis $F_0(t_0) = \theta$, respectively, and $\mathbb{D}$ has a distribution, characterized in Banerjee and Wellner (2001) (see Theorem 2.5 in that paper). No analytic information is available for the distribution of $\mathbb{D}$. However, the 95% percentile of the distribution of $\mathbb{D}$ was determined by simulation to be approximately 2.26916

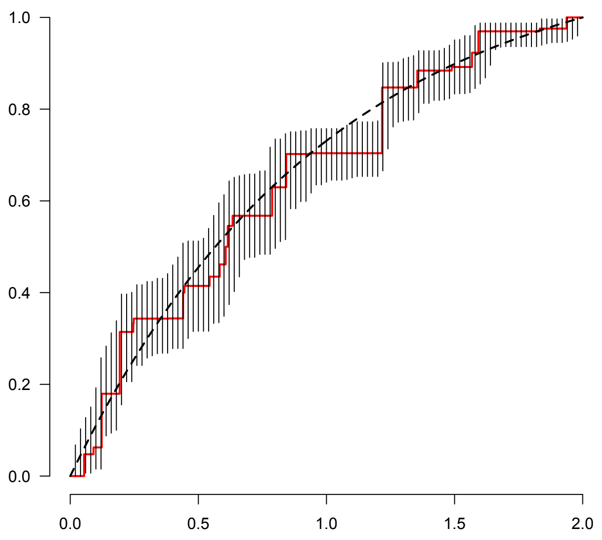


Fig 10: Banerjee-Wellner confidence intervals for the (truncated exponential) distribution function $F_0$, based on a sample of 1000 observations $(T_i, \Delta_i)$, where $T_i$ is Uniform$(0, 2)$. The intervals are based on 1000 (smooth) bootstrap samples of indicators $\Delta_i^*$. The dashed curve is the truncated exponential distribution function.

Pictures of the proportion of non-coverages of the real value of the distribution in 1000 samples, comparable to Figure 7, are shown in Figure 11. The coverage is again somewhat off at the boundary, but less so than the Sen-Xu intervals. However, note that the bias-corrected confidence intervals based on the SMLE are also well-behaved at the boundary.

The pictures can be reproduced by running the R script CI_Banerjee_Wellner_curstat.R in the directory CI_Banerjee_Wellner_simulation, which, in turn, belongs to the directory Confidence Intervals (current status simulations). It takes about 35 minutes on my laptop.

## 6. Acknowledgements

I want to thank Richard Gill for testing several `Rcpp` algorithms on Mac, Windows and Linux. Harold Hvistendahl tested the original `C++` program for bootstrapping on Windows (where it seemed to work fine)

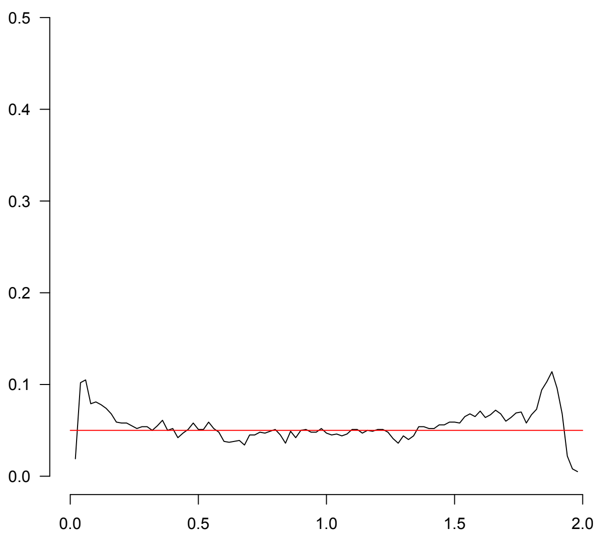


Fig 11: Proportion of times that $F_0(t_i)$, $t_i = 0.02, 0.04, \dots$ is not in the CI's in 1000 samples $(T_1, \Delta_1) \dots, (T_n, \Delta_n)$ for the Banerjee-Wellner intervals.

and the original `Rcpp` version of this (which first crashed in all environments). He also added some text to the output, which seemed a good idea. Dimitri Tischenko spotted a bug in the first `Rcpp` version, using Valgrind. After repairing this bug, we did not experience crashes any more. Frank van der Meulen tested the GUI application on his MacBook Pro with system 10.10. Bodhi Sen elucidated a point in his paper Sen and Xu (2015) to me. I feel very grateful for their contributions.

## References

Banerjee, M. and Wellner, J. A. (2001). Likelihood ratio tests for monotone functions. *Ann. Statist.* **29** 1699–1731. MR1891743 (2003c:62072)

Banerjee, M. and Wellner, J. A. (2005). Confidence intervals for current status data. *Scand. J. Statist.* **32** 405–424. MR2204627

Eddelbuettel, D. (2013). *Seamless R and C++ Integration with Rcpp*. Springer, New York.

Groeneboom, P. (2013). Nonparametric (smoothed) likelihood and integral equations.

Groeneboom, P., Jongbloed, G. and Wellner, J. A. (2008). The support reduction algorithm for computing nonparametric function estimates in mixture models. *Scand. J. Statist.* **35** 385–399. MR2446726 (2009m:62115)

Groeneboom, P., Jongbloed, G. and Witte, B. I. (2010). Maximum smoothed likelihood estimation and smoothed maximum likelihood estimation in the current status model. *Ann. Statist.* **38** 352–387.

Groeneboom, P. and Jongbloed, G. (2014). *Nonparametric Estimation under Shape Constraints*. Cambridge Univ. Press, Cambridge.

Groeneboom, P. and Jongbloed, G. (2015). Nonparametric confidence intervals for monotone functions. To appear in Annals of Statistics.

Groeneboom, P., Maathuis, M. H. and Wellner, J. A. (2008). Current status data with competing risks: consistency and rates of convergence of the MLE. *Ann. Statist.* **36** 1031–1063. MR2418648 (2009h:62039)

Kitayaporn, D., Vanichseni, S., Mastro, T. D., Raktham, S., Vaniyapongs, T., Des Jarlais, D. C., Wasi, C., Young, N. L., Sujarita, S., Heyward, W. L. and Esparza, J. (1998). Infection with HIV-1 subtypes B and E in injecting drug users screened for enrollment into a prospective cohort in Bangkok, Thailand. *J. Acquir. Immune Defic. Syndr. Hum. Retrovirol.* **19** 289–295.

Kosorok, M. R. (2008). Bootstrapping the Grenander estimator. In *Beyond parametrics in interdisciplinary research: Festschrift in honor of Professor Pranab K. Sen. Inst. Math. Stat. Collect.* **1** 282–292. Inst. Math. Statist., Beachwood, OH.

Li, C. and Fine, J. P. (2013). Smoothed nonparametric estimation for current status competing risks data. *Biometrika* **100** 173–187. MR3034331

Maathuis, M. H. (2013). MLEcens. R package. Version 0.1-4.

Maathuis, M. H. and Hudgens, M. G. (2011). Nonparametric inference for competing risks current status data with continuous, discrete or grouped observation times. *Biometrika* **98** 325–340. MR2806431 (2012e:62108)

Press, W. H., Teukolsky, S. A., Vetterling, W. T. and Flannery, B. P. (1992). *Numerical recipes in C*, Second ed. Cambridge University Press, Cambridge The art of scientific computing. MR1201159 (93i:65001b)

Sen, B., Banerjee, M. and Woodroofe, M. B. (2010). Inconsistency of bootstrap: the Grenander estimator. *Ann. Statist.* **38** 1953–1977. MR2676880 (2011f:62046)

Sen, B. and Xu, G. (2015). Model based bootstrap methods for interval censored data. *Comput. Statist. Data Anal.* **81** 121–129. MR3257405

Vanichseni, S., Kitayaporn, D., Mastro, T. D., Mock, P. A., Raktham, S., C., D. J. D., Sujarita, S., Srisuwanvilai, L. O., Young, N. L., Wasi, C., Subbarao, S., Heyward, W. L., Esparza, L. and Choopanya, K. (2001). Continued high HIV-1 incidence in a vaccine trial preparatory cohort of injection drug users in Bangkok, Thailand. *AIDS* **15** 397–405.