Recurrent events with R (Part I)

Juan R Gonzalez

Contents

1 Introduction

Objectives

- Understand the concept of survival analysis with recurrent events data
- Learn how to estimate survival function in recurrent event settings
- Know how to compare survival curves between groups

2 Estimating survival function

The library implementing the models described in the theoretical part is called `survrec`. It can be installed into R by typing:

```
devtools::install_github("isglobal-brge/survrec")
```

Then, the package is loaded in R as usually:

```
library(survrec)
```

Data for illustrating purposes belong to a study about hospital readmissions. They can be loaded by:

```
data(colon, package="survrec")
head(colon)
##      hc time event chemoter dukes distance
## 1  5634   24     1        2     3        1
## 2  5634  433     1        2     3        1
## 3  5634  580     0        2     3        1
## 4 10767  489     1        1     2        1
## 5 10767  693     0        1     2        1
## 6 15843   15     1        1     2        1
```

Here, we observe as repeated measurements are in different rows. The variabel *hc* encodes the identification number (in that case *hc* stands for the `clinical recordnumber`) to indicate that this individula is having several observations. For instance, the individual with *hc* number 5346 is having two hospital readmissions at time 24 and 433 ($event = 1$) and then the patient is followed up intiltime 580 without having any other hospital admission ($event = 0$).

As in the case of estimating survival function in standard survival analysis, an object encoding survival time has to be created. The object `Survr` is similar to the one created in `survrec` with the function `Surv`. This function requires the time variable, the censored variable and a variable indicating the repeated events that must have a censored time in the last observation. If this is not happening an error message is obtained

```
> with(colon, Survr(time, event, hc))
Error en Survr(time, event, hc) :
Data doesn't match. Every subject must have a censored time
```

If this error is observed, a censored time to each individual has to be added. If the final of the study corresponds o the last event, this value must be 0. This can automatically be creaed by using the function `addCenTime` of library `gcmrec` that can be installed from GitHub using `devtools` package:

```
devtools::install_github("isglobal-brge/gcmrec")
```

```
library(gcmrec)
colon2 <- addCenTime(colon)
head(with(colon2, Survr(hc, time, event)))
## [1]  5634  5634  5634 10767 10767 15843
```

NOTE: the function `addCenTime` has by default the arguments `id=1`, `time=2` and `event=3` indicating the columns where those variables are located in your `data.frame`.

2.1 Pena-Strawderman-Hollander (PSH) estimator

The estimation of survival curve using Pena-Strawderman-Hollander method is estimated by

```
ans.psh <- survfitr(Survr(hc, time, event) ~ 1, data=colon2,
                    type="pena-strawderman-hollander")
ans.psh
## Survival for recurrent event data
##     n events mean se(mean) median recurrences: min max median
##   403    458  912     36.2    436                0  22      1
```

Notice that this estimation is similar to Kaplan-Meier, byt the estimation of the standard error is different since PSH considers that times are correlated

```
library(survival)
ans.km <- survfit(Surv(time, event) ~ 1, data=colon2)
all.equal(summary(ans.km)$surv, ans.psh$surv)
## [1] TRUE
head(cbind(ans.km$std, ans.psh$std))
##             [,1]        [,2]
## [1,] 0.000000000 0.003370111
## [2,] 0.003416111 0.005770836
## [3,] 0.005976608 0.007474231
## [4,] 0.007905453 0.007862375
## [5,] 0.008355873 0.008652295
## [6,] 0.009305638 0.008898736
```

2.2 Frailty model estimator (FRMLE)

Pena-Strawderman-Hollander also proposed to estimate the survival function with recurrent event data when inter-occurrence times are correlate by using a Frailty model. This model can be estimated by executing

```
ans.fra <- survfitr(Survr(hc, time, event) ~ 1, data=colon2,
                    type="MLEfrailty")
##
## Needs to Determine a Seed Value for Alpha
##  Seed Alpha:  0.5
##
##  Alpha estimate= 1.044693
##
ans.fra
## Survival for recurrent event data
##     n events mean se(mean) median recurrences: min max median
##   403    458 1152     47.1   1088                  0  22      1
```

A large *alpha* value would indicate that the variance of the frailty is almost 0 (*alpha* encodes the precission of the gamma distribution by using the formulation presented in PSH). In other words, if *alpha* is 0, the independent model (e.g PSH) is enough to fit the data. This can be visually check by comparing both survival curves as we illustrate in the next section.

2.3 Wand-Chang estimator (WC)

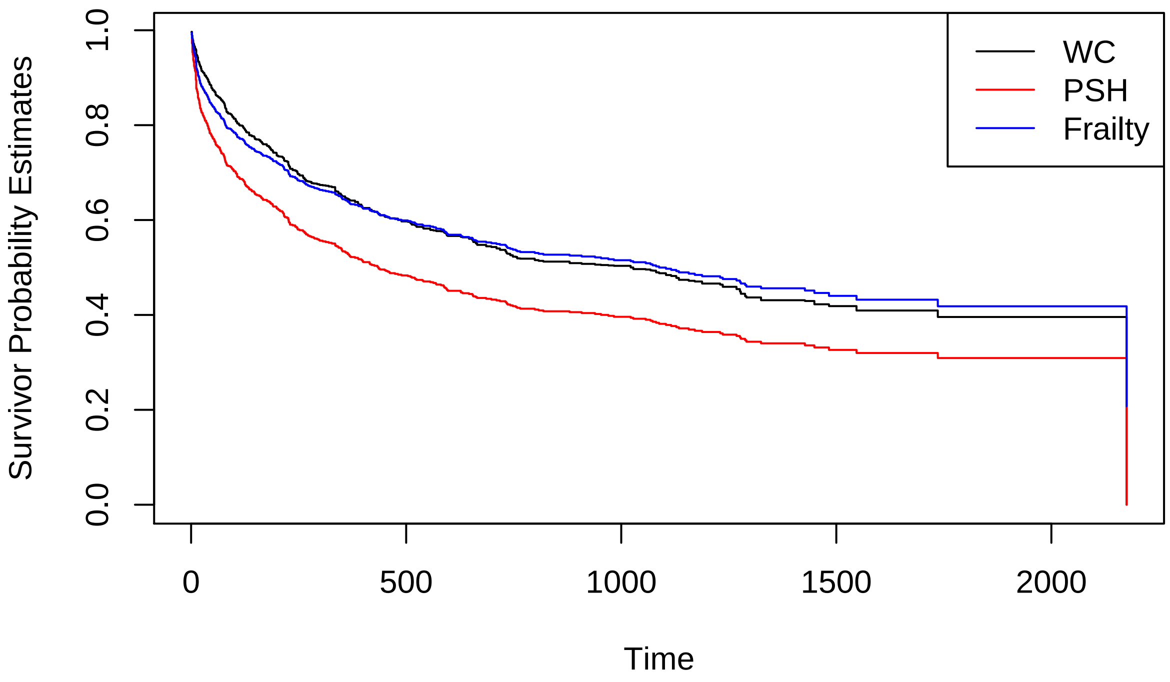
The method proposed by Wang and Chang can be estimated using

```
ans.wc <- survfitr(Survr(hc, time, event) ~ 1, data=colon2,
                    type="wang-chang")
ans.wc
## Survival for recurrent event data
##     n events mean se(mean) median recurrences: min max median
##   403    861 1130     49.9   1028                  1  23      2
```

3 Estimator selection

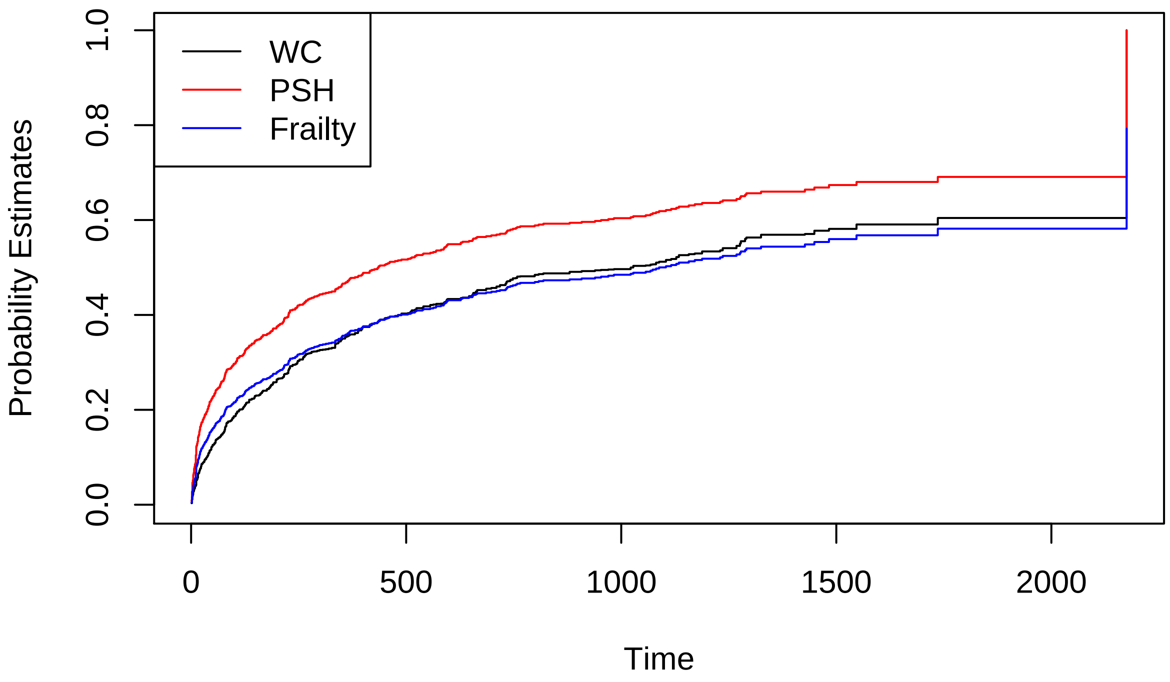
PSH method assumes that all inter-ocurrence times are independent, while FRMLE assumes that data are correlated and estimates such correlation by using a Frailty model. WC estimator can capture both situations. Therefore, one can decide which is the best model by comparing the survival curves estimated by using the three approaches.

```
plot(ans.wc, conf.int=FALSE)
lines(ans.psh, col="red")
lines(ans.fra, col="blue")
legend("topright", c("WC", "PSH", "Frailty"), col=c("black", "red", "blue"), lty=1)
```



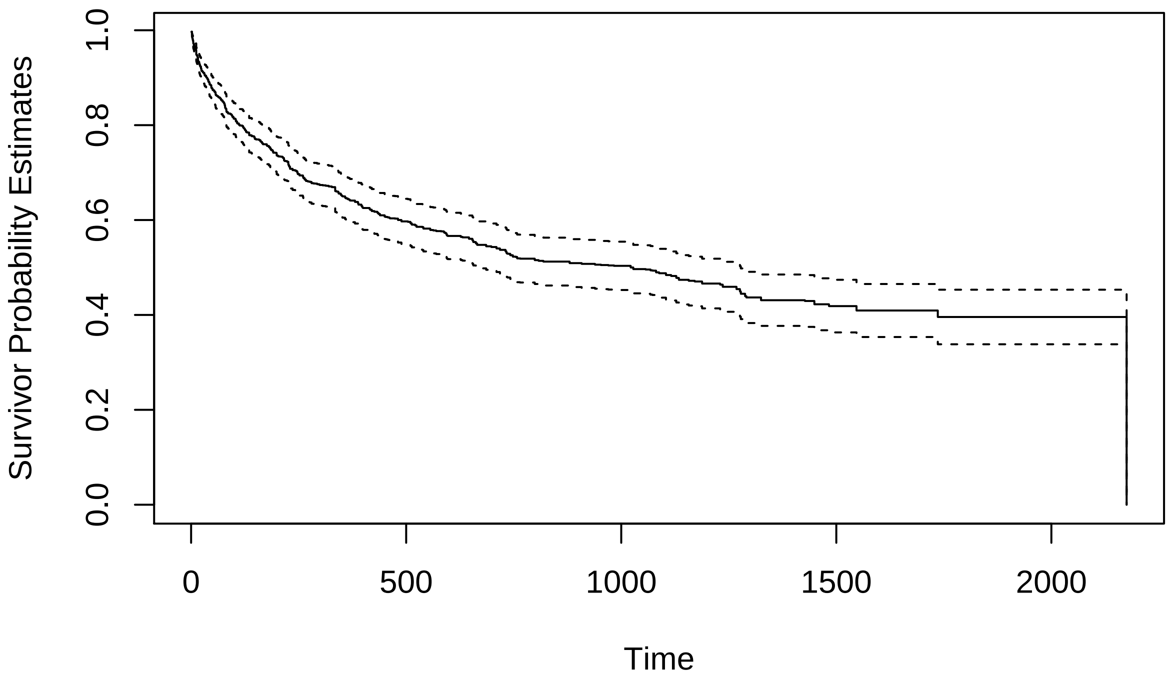
In this plot, we can observe as FRLME and WC estimators are completely diffenrent of PSH. This indicates that data are correlated and that PSH model is underestimating the real survival function. Notice that the probability distribution function can also be estimating by changing the argument prob=TRUE.

```
plot(ans.wc, conf.int=FALSE, prob=TRUE)
lines(ans.psh, prob=TRUE, col="red")
lines(ans.fra, prob=TRUE, col="blue")
legend("topleft", c("WC", "PSH", "Frailty"), col=c("black", "red", "blue"), lty=1)
```



Confidence bands are depicted by default

```
plot(ans.wc)
```



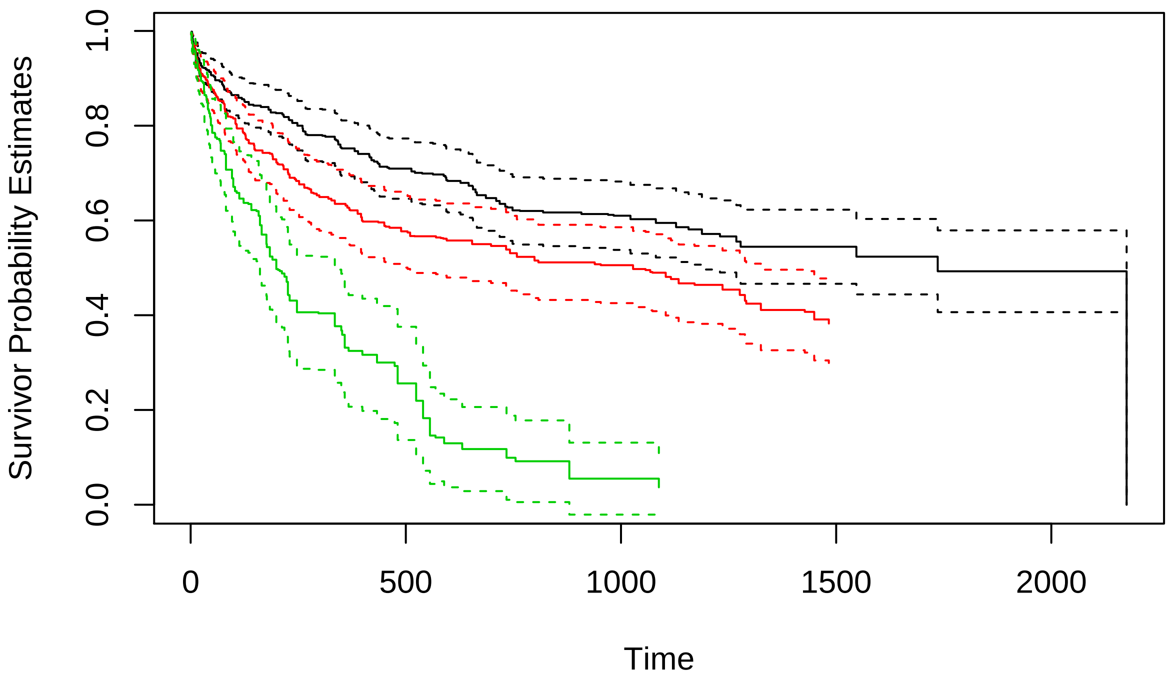
The methodology previously described is further explained in a review written in Spanish by Gonzalez and Pena which '.pdf' is also available in the teaching course material repository.

4 Comparing survival curves

4.1 Pointwise bootstrap confidence intervals

Let us imagine that we are interested in comparing survival curves accross groups. For example, in our datasets one may observe differences between Duke stage that encodes how advanced is the tumor:

```
fit.dukes <- survfitr(Survr(hc, time, event) ~ as.factor(dukes), data=colon2,
                      type="wang-chang")
plot(fit.dukes)
```



In the single setting, there are methods to compare the behaviour of the overall curve such as the log-rank or Wilcoxon tests among others. In recurrent event settings, only there are methods to compare survival curves at a given time. These methods are describe in Gonzalez, Pena and Delicado and are based on bootstrap techniques. This is performed by using the `survdiffr()` function. The argument `q=0.5` indicates that we are interested in estimating confidence interval of median survival time:

```
fit <- survdiffr(Survr(hc, time, event) ~ as.factor(dukes), data=colon2, q=0.7)
```

Confidence intervals of median survival time to the first group of variable *dukes* is the computed by

```
boot.ci(fit$"1")
## BOOTSTRAP CONFIDENCE INTERVAL CALCULATIONS
## Based on 500 bootstrap replicates
##
## CALL :
## boot.ci(boot.out = fit$"1")
##
## Intervals :
## Level      Normal              Basic
## 95%   (275.7, 789.0 )   (328.0, 733.0 )
##
## Level     Percentile            BCa
## 95%   (343.0, 748.0 )   (342.0, 733.2 )
## Calculations and Intervals on Original Scale
```

4.2 Comparing the whole survival curve

Martinez et al proposed a method to test

$$\begin{aligned} \mathrm{H}_0 &: S_1(t) = S_2(t) \\ \mathrm{H}_1 &: S_1(t) \neq S_2(t) \end{aligned} \tag{1}$$

in the recurrent event settings. The methods are implemented in the package `TestSurvRec` that can be install into R by

```
devtools::install_github("isglobal-brge/TestSurvRec")
```

5 Exercise (to deliver)

Data `lymphoma` is available at `gcmrec` package. It contains cancer relapses times after first treatment in patients diagnosed with low grade lymphoma. Data can be loaded into R by executing

```
data(lymphoma, package = "gcmrec")
```

NOTE: variable *time* contains inter-occurrence times, *event* is the censoring variable that is 1 for cancer relapses and 0 for the last follow up time indicating that the event is not observed and the variable *id* identifies each patient.

Exercise 1:

- Estimate survival function using PeÃ±a-Strawderman-Hollander, Wang-Chang and a Frailty model.
- Represent the three estimated survival curves in a figure.
- Is there correlation among inter-ocurrence times?
- Which is the best method to analyze these data?

Exercise 2:

- Investigate how the package `TestSurvRec` compares two whole survival curves (see References section to both package and manuscript describing how it works).
- By using this method:
  - Compare cancer relapse times between males and females (variable *sex*).
  - Compare cancer relapse time between patiens having single lesions and localized lesions (variable *distrib*)
  - Compare cancer relapse time between patiens having single lesions and lesions in more than one nodal site (variable *distrib*)

NOTE: variable *distrib* encodes the lesions involved at diagnosis and has 4 categories (0=Single, 1=Localized, 2=More than one nodal site, 3=Generalized)

---

6 References

- The `survrec` package (https://github.com/isglobal-brge/survrec)
- The `gcmrec` package (https://github.com/isglobal-brge/gcmrec)
- The `TestSurvRec` package (https://github.com/isglobal-brge/TestSurvRec)
- Pena, E.A., Strawderman, R. and Hollander, M. (2001). Nonparametric Estimation with Recurrent Event Data. J. Amer. Statist. Assoc 96, 1299-1315.
- Wang, M.C. and Chang, S.H. (1999). Nonparametric Estimation of a Recurrent Survival Function. J. Amer. Statist. Assoc 94, 146-153.

- Gonzalez, J.R and Pena, E. (2004) Estimacion no parametrica de la funcion de superviencia para datos con eventos recurrentes. Rev. Esp. Salud Publica 78(2). Available here.
- Gonzalez, J.R., Pena, E. and Delicado, P. (2010) Confidence intervals for median survival time with recurrent event data. Computational Statistics and Data Analysis, 54 (1) 78-89.
- Martinez C., Ramirez, G., Vasquez M. (2009). Pruebas no parametricas para comparar curvas de supervivencia de dos grupos que experimentan eventos recurrentes. Propuestas. Revista Ingenieria U.C., Vol 16, 3, 45-55. Available here.

7 Session information

```
## R version 3.4.1 (2017-06-30)
## Platform: x86_64-w64-mingw32/x64 (64-bit)
## Running under: Windows 10 x64 (build 16299)
##
## Matrix products: default
##
## locale:
## [1] LC_COLLATE=Spanish_Spain.1252  LC_CTYPE=Spanish_Spain.1252
## [3] LC_MONETARY=Spanish_Spain.1252 LC_NUMERIC=C
## [5] LC_TIME=Spanish_Spain.1252
##
## attached base packages:
## [1] stats     graphics  grDevices utils     datasets  methods   base
##
## other attached packages:
## [1] gcmrec_1.0-5    survival_2.41-3 survrec_1.2-5   boot_1.3-19
## [5] knitr_1.20      BiocStyle_2.4.1
##
## loaded via a namespace (and not attached):
##  [1] Rcpp_0.12.12    lattice_0.20-35 digest_0.6.12   rprojroot_1.3-2
##  [5] grid_3.4.1      backports_1.1.0 magrittr_1.5    evaluate_0.10.1
##  [9] stringi_1.1.6   Matrix_1.2-10   rmarkdown_1.8   splines_3.4.1
## [13] tools_3.4.1     stringr_1.3.0   yaml_2.1.16     compiler_3.4.1
## [17] htmltools_0.3.6
```