A wrapper routine for MLE estimation in parametric regression models for survival analysis in R

maxlogLreg

Jaime Mosquera; Freddy Hernández

November 6, 2019

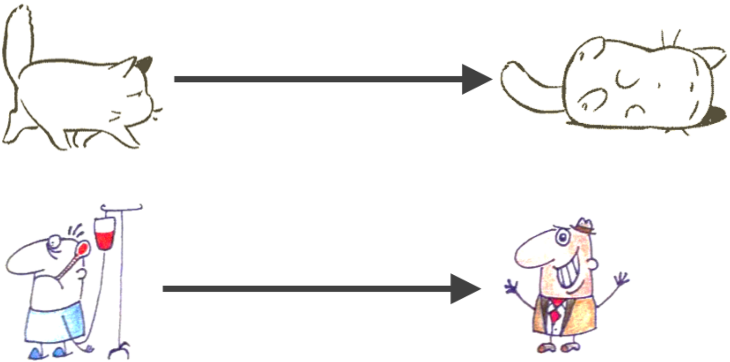
UNIVERSIDAD NACIONAL DE COLOMBIA

XXIV CONGRESSO SOCIEDADE PORTUGUESA DE ESTATÍSTICA

Section 1

Survival Analysis

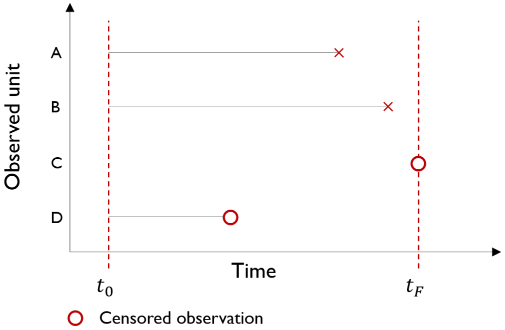
About Survival Analysis



One directional transitions, from A to B (irreversible).

Asumptions

- Observed events.
- Non observed events:
  - Right censorship (mainly).
  - Left censorship (also available).



Linear regression models

A linear regression model can be expressed as follows:

$$y_i \stackrel{\text{iid.}}{\sim} \mathcal{D}(\boldsymbol{\theta}) \qquad (1)$$

$$\theta_j = \mathbf{X}_j^\top \boldsymbol{\beta}_j, \qquad (2)$$

- y_i is a response (random) variable with $i = 1, 2, ..., n$ observations.
- $\mathcal{D}$ a distribution with k parameters.
- $\boldsymbol{\theta} = (\theta_1, ..., \theta_k)$ is the vector of the distribution parameters.
- $\boldsymbol{\beta}_j = (\boldsymbol{\beta}_1, \boldsymbol{\beta}_2, ..., \boldsymbol{\beta}_k)$ are the fixed effects vector of covariates of j^{th} distribution parameter.
- $\mathbf{X}_j$ is a known design matrix of order $n \times k$.

Section 2

Estimation

MLE

Mathematical principle behind estimation in this models:

$$\hat{\beta}_j = \underset{\beta_j \in B}{\arg\max} \left\{ \log \left[\prod_{i=1}^{n} f(y_i)^{a_i} S(y_i)^{r_i} F(y_i)^{l_i} \right] \right\}, \tag{3}$$

$f(\cdot), F(\cdot)$ and $S(\cdot)$ are de probability density, cumulative density and survival functions.

$$a_i = \begin{cases} 1, & \text{if } y_i \text{ is not censored.} \\ 0, & \text{in other case} \end{cases} \tag{4}$$

and so on for left and right censorship with r_i and l_i respectively.

Section 3

Our routine

How to get it?

- Download and install `EstimationTools` package.

```r
if (!require('devtools')) install.packages('devtools')
devtools::install_github('Jaimemosg/EstimationTools',
                         force = TRUE)
library(EstimationTools)
```

- Repo in GitHub: https://github.com/Jaimemosg/EstimationTools
- GitHub web site:https://jaimemosg.github.io/EstimationTools/
- CRAN repo:
  https://cran.r-project.org/web/packages/EstimationTools/index.html

Repo

build passing | build passing | CRAN 1.2.1 – 12 days ago | repo status Active | downloads 289/month | licence GPL-3

EstimationTools

The goal of `EstimationTools` is to provide a routine for parameter estimation of probability density/mass functions in `R`.

Installation

You can install the lastest version of `EstimationTools` typing the following command lines in `R` console:

```
if (!require('devtools')) install.packages('devtools')
devtools::install_github('Jaimemosg/EstimationTools', force = TRUE)
library(EstimationTools)
```

Or you can install the released version from CRAN if you prefer. You can also type the following command lines in `R` console:

```
install.packages("EstimationTools")
```

You can visit the package website to explore the vignettes (articles) and functions reference.

EstimationTools 2.0.0 | Reference | Articles

Reference

All functions

`Fibers`	Tensile strengths
`logit_link()`	Logit link function (for estimation with `maxlogL` object)
`log_link()`	Logarithmic link function (for estimation with `maxlogL` object)
`maxlogL()`	Maximum Likelihood Estimation for parametric distributions
`maxlogLreg()`	Maximum Likelihood Estimation for parametric linear regression models
`NegInv_link()`	Negative inverse link function (for estimation with `maxlogL` object)
`summary(<maxlogL>)`	Summarize Maximum Likelihood Estimation

Main arguments

```r
maxlogLreg(formulas, y_dist, data, fixed, link,
           start, lower, upper, ...)
```

- `formulas`: a list with formula objects to specify linear predictors of each parameter.
- `y_dist`: a formula to define the response variable and its distribution.
- `fixed`: a list stating the known parameters.
- `link`: a list to define link functions application.
- `start, lower, upper`: lists to set out initial values, lower and upper bounds for search.

Section 4

Motivation: usual approach vs. `maxlogLreg`

Application example # 1

Lets assume a failure test of 500 hours of 20 electronic devices. Only 10 of these failures have been observed (those with $Status = 1$).

Table 1: Failure times in the test.

Time	Status	Time	Status
55	1	500	0
187	1	500	0
216	1	500	0
240	1	500	0
244	1	500	0
335	1	500	0
361	1	500	0
373	1	500	0
375	1	500	0
386	1	500	0

Data from itl.nist.gov

Application example # 1

Selected distribution:

$$f(y|\alpha, k) = \frac{\alpha}{k}\left(\frac{y}{k}\right)^{\alpha-1} \exp\left[-\left(\frac{x}{k}\right)^{\alpha}\right]$$

The model:

$$\begin{aligned} y_i &\overset{\text{iid.}}{\sim} WEI(\alpha,\ k), \\ \alpha &= c_1, \\ k &= c_2, \end{aligned}$$

c_1 and c_2 are unknown parameters.

Application example # 1

Code implementation:

```
logL <- function(theta){
  beta <- theta[1]
  eta <- theta[2]
  logf <- dweibull(x = y[,1], shape = beta, scale = eta,
                   log = TRUE)
  logS <- log(pweibull(q = y[,1], shape = beta, scale = eta,
                       lower.tail = FALSE))
  l <- sum( logf*y[,2] + logS*(1 - y[,2]) )
  return(-l)
}
fit <- nlminb(start = c(1,100), objective = logL, lower = c(0,0))
fit$par
```

```
## [1]   1.725626 606.004873
```

With our routine

```r
formulas <- list(scale.fo = ~ 1, shape.fo = ~ 1)

start <- list(
  scale = list(Intercept = 100),
  shape = list(Intercept = 10)
)
lower <- list(
  scale = list(Intercept = 0),
  shape = list(Intercept = 0)
)

mod_weibull <- maxlogLreg(formulas,
                          y_dist = Surv(times, status) ~ dweibull,
                          start = start,
                          lower = lower)
```

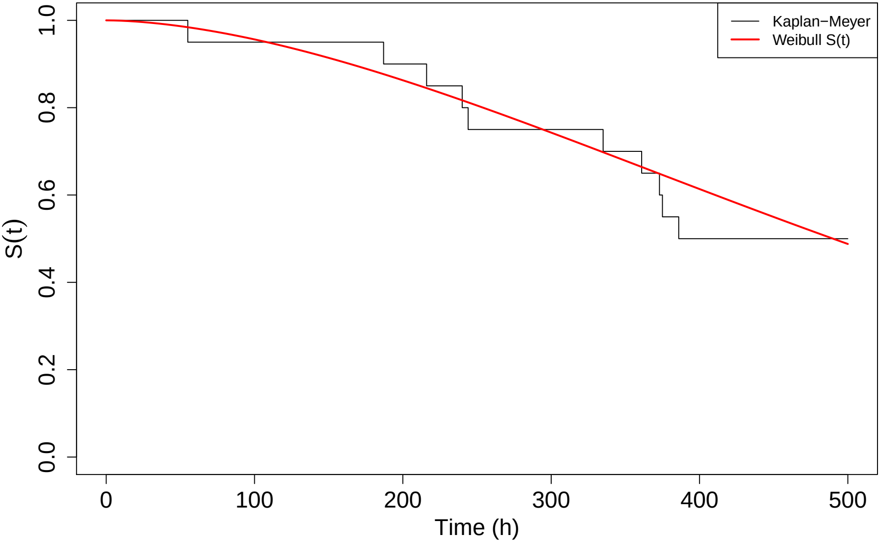
With our routine

The user does not have to write loglikelihood again and again.

```r
summary(mod_weibull)
```

```
## ---------------------------------------------------------------
## Optimization routine: nlminb
## Standard Error calculation: Hessian from optim
## ---------------------------------------------------------------
##        AIC      BIC
##   154.2437 156.2352
## ---------------------------------------------------------------
## Fixed effects for g(shape)
## ---------------------------------------------------------------
##             Estimate Std. Error Z value  Pr(>|z|)
## (Intercept)   1.7256     0.5034  3.4279 0.0006083 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
## ---------------------------------------------------------------
## Fixed effects for g(scale)
## ---------------------------------------------------------------
##             Estimate Std. Error Z value  Pr(>|z|)
## (Intercept)   606.00     124.43  4.8703 1.114e-06 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
## ---------------------------------------------------------------
## Note: p-values valid under asymptotic normality of estimators
## ---
```

The fit



Kaplan−Meyer
Weibull S(t)
S(t)
0.0
0.2
0.4
0.6
0.8
1.0
0
100
200
300
400
500
Time (h)

Section 5

Implementation of a new regression model

Application example # 2

Neck cancer data

The data corresponds to a randomized clinical trial to compare two therapies for head and neck cancer (Khan 2018; Efron 1988):

i. 51 patients treated with radiation only.
ii. 45 patients treated with radiation plus chemotherapy.
iii. Response: survival times in days.

We fitted the Weibull model (usually used) and Exponentiated Weibull (EW) model.

The regression models

- Weibull:

$$f(t|\rho,\kappa) = \kappa\rho\,(\rho t)^{\kappa-1}\exp\left[-(\rho t)^{\kappa}\right]$$

$$\begin{aligned} T_i &\overset{\text{iid.}}{\sim} WEI2(\rho,\ \kappa), \\ \log(\rho) &= \beta_0 + \beta_1 x_1, \\ \log(\kappa) &= \alpha_0, \end{aligned}$$

- Exponentiated Weibull:

$$g(t|\rho,\kappa) = \kappa\rho\gamma\,(\rho t)^{\kappa-1}\exp\left[-(\rho t)^{\kappa}\right]\left\{1-\exp\left[-(\rho t)^{\kappa}\right]\right\}$$

$$\begin{aligned} T_i &\overset{\text{iid.}}{\sim} EW(\rho,\ \kappa,\ \gamma), \\ \log(\rho) &= \beta_0 + \beta_1 x_1, \\ \log(\kappa) &= \alpha_0, \\ \log(\gamma) &= \lambda_0, \end{aligned}$$

Weibull regression

```
## --------------------------------------------------------------
## Optimization routine: nlminb
## Standard Error calculation: Hessian from optim
## --------------------------------------------------------------
##        AIC      BIC
##   1082.519 1090.212
## --------------------------------------------------------------
## Fixed effects for g(rho)
## --------------------------------------------------------------
##             Estimate Std. Error  Z value  Pr(>|z|)
## (Intercept) -6.82473    0.21110 -32.3294 < 2.2e-16 ***
## Therapy      0.78603    0.27880   2.8193  0.004813 **
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
## --------------------------------------------------------------
## Fixed effects for g(k)
## --------------------------------------------------------------
##             Estimate Std. Error Z value Pr(>|z|)
## (Intercept) -0.16173    0.09210  -1.756  0.07909 .
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
## --------------------------------------------------------------
## Note: p-values valid under asymptotic normality of estimators
## ---
```

Estimates by Khan: $\hat{\beta}_0 = -6.825,\ \hat{\beta}_1 = 0.786,\ \hat{\alpha}_0 = 0.162$

EW regression

Implement the distribution

```
dEW <- function(x, rho, k, gam, log = FALSE) {
  prod1 <- -(rho*x)^k
  prod2 <- (rho*x)^(k-1)
  f <- k*rho*gam*prod2*exp(prod1)*( -expm1(prod1) )^(gam-1)
  if (log == FALSE)
    density <- f
  else density <- log(f)
  return(density)
}

pEW <- function(q, rho, k, gam, lower.tail = TRUE, log.p = FALSE) {
  prod <- -(rho*q)^k
  cdf <- (-expm1(prod))^gam
  if (lower.tail == TRUE)
    cdf <- cdf
  else cdf <- 1 - cdf
  if (log.p == FALSE)
    cdf <- cdf
  else cdf <- log(cdf)
  cdf
}
```

EW regression

```r
formulas <- list(rho.fo = ~ Therapy, k.fo = ~ 1, gam.fo = ~ 1)

start <- list(
  rho = list(Intercept = 7, Therapy = 2),
  k = list(Intercept = -4), gam = list(Intercept = 4)
)
lower <- list(
  rho = list(Intercept = 0, Therapy = 0),
  k = list(Intercept = -6), gam = list(Intercept = 0)
)
upper <- list(
  rho = list(Intercept = 20, Therapy = 5),
  k = list(Intercept = 5), gam = list(Intercept = 10)
)

control <- list(step.max = 1e-5, step.min=1e-7, eval.max=500,
                iter.max=500, rel.tol=1e-8)

modEW <- maxlogLreg(formulas, start = start, control = control,
                    lower = lower, upper = upper,
                    y_dist = Surv(Time, status) ~ dEW,
                    link = list(over = c("rho","k","gam"),
                                fun = rep("log_link", 3)))
```

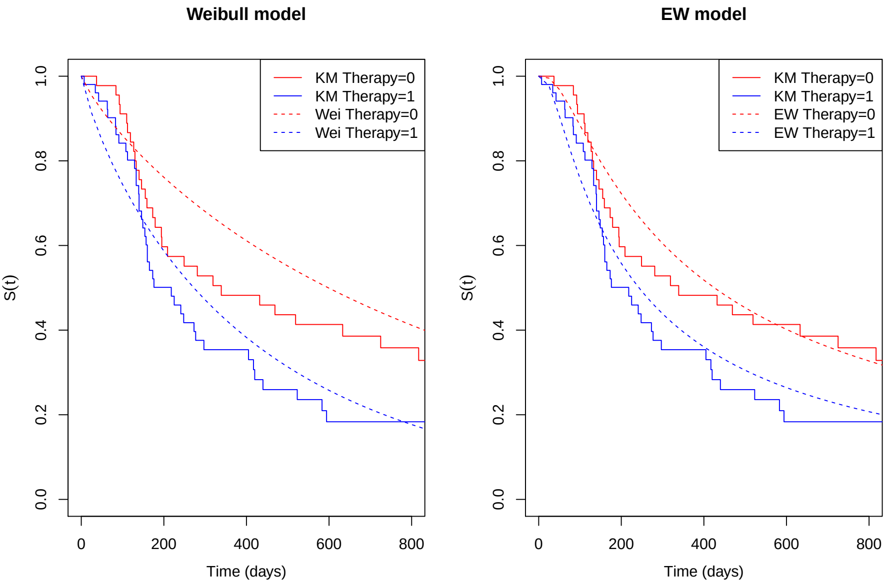
EW regression

```r
summary(modEW)
```

```
## -----------------------------------------------------------------
## Optimization routine: nlminb
## Standard Error calculation: Hessian from optim
## -----------------------------------------------------------------
##        AIC      BIC
##   1064.556 1074.813
## -----------------------------------------------------------------
## Fixed effects for g(rho)
## -----------------------------------------------------------------
##             Estimate Std. Error Z value Pr(>|z|)
## (Intercept) 10.17474   15.33130  0.6637  0.50688
## Therapy      0.56092    0.26730  2.0985  0.03586 *
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
## -----------------------------------------------------------------
## Fixed effects for g(k)
## -----------------------------------------------------------------
##             Estimate Std. Error Z value Pr(>|z|)
## (Intercept)  -2.1157     0.6429 -3.2908 0.000999 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
## -----------------------------------------------------------------
## Fixed effects for g(gam)
## -----------------------------------------------------------------
##             Estimate Std. Error Z value Pr(>|z|)
## (Intercept)   6.7060     4.1885  1.6011   0.1094
## -----------------------------------------------------------------
## Note: p-values valid under asymptotic normality of estimators
## ---
```

Estimates by Khan: $\hat{\beta}_0 = 10.183,\ \hat{\beta}_1 = 0.561,\ \hat{\alpha}_0 = 2.116,\ \hat{\lambda}_0 = 6.708$

Comparison



Weibull model
EW model
S(t)
1.0
0.8
0.6
0.4
0.2
0.0
0
200
400
600
800
Time (days)
KM Therapy=0
KM Therapy=1
Wei Therapy=0
Wei Therapy=1
EW Therapy=0
EW Therapy=1

Comparison

```r
AIC(modW, modEW)
```

```
##       df      AIC
## modW   3 1082.519
## modEW  4 1064.556
```

The AIC decreases, so EW model is better than Weibull model.

Conclusions

- Our routine performs a successsfully estimation process.
- `maxlogLreg` facilitates the definition of initial values and bounds in estimation for the final user.
- This function is flexible: allows any distribution.
- Initial values are always a difficult issue.

References

Efron, Bradley. 1988. "Logistic Regression, Survival Analysis, and the Kaplan-Meier Curve." *Journal of the American Statistical Association* 83 (402): 414. https://doi.org/10.2307/2288857.

Khan, Shahedul A. 2018. "Exponentiated Weibull regression for time-to-event data." *Lifetime Data Analysis* 24 (2): 328–54. https://doi.org/10.1007/s10985-017-9394-3.

Mullen, Katharine, David Ardia, David Gil, Donald Windover, and James Cline. 2011. "DEoptim : An R Package for Global Optimization by Differential Evolution." *Journal of Statistical Software* 40 (6). https://doi.org/10.18637/jss.v040.i06.

Nash, John C. 1979. *Compact Numerical Methods for Computers. Linear Algebra and Function Minimisation*. 2nd Editio. Bristol: Adam Hilger.

Pawitan, Yudi. 2013. *In all likelihood: statistical modelling and inference using likelihood*. Oxford University Press.