

# Package ‘emplik’

August 17, 2018

**Maintainer** Mai Zhou <mai@ms.uky.edu>

**Version** 1.0-4.3

**Depends** R (>= 3.0)

**Imports** quantreg, stats

**Suggests** KMsurv, boot

**Author** Mai Zhou. (Art Owen for el.test(). Yifan Yang for C code.)

**Description** Empirical likelihood ratio tests for means/quantiles/hazards from possibly censored and/or truncated data. Now does regression too. This version contains some C code.

**Title** Empirical Likelihood Ratio for Censored/Truncated Data

**License** GPL (>= 2)

**URL** <http://www.ms.uky.edu/~mai/EmpLik.html>

**NeedsCompilation** yes

**Repository** CRAN

**Date/Publication** 2018-08-17 20:50:54 UTC

## R topics documented:

BJoint . . . . .	2
bjtest . . . . .	3
bjtest1d . . . . .	5
bjtestII . . . . .	6
el.cen.EM . . . . .	8
el.cen.EM2 . . . . .	10
el.cen.test . . . . .	14
el.ltrc.EM . . . . .	16
el.test . . . . .	18
el.test.wt . . . . .	20
el.test.wt2 . . . . .	21
el.trun.test . . . . .	23
emplikH.disc . . . . .	25

emplikH.disc2 . . . . .	27
emplikH1.test . . . . .	29
emplikH2.test . . . . .	31
emplikHs.disc2 . . . . .	33
emplikHs.test2 . . . . .	35
findUL . . . . .	38
findUL2 . . . . .	40
findULold . . . . .	42
myeloma . . . . .	43
RankRegTest . . . . .	44
RankRegTestH . . . . .	46
ROCnp . . . . .	48
ROCnp2 . . . . .	49
smallcell . . . . .	51
WRegEst . . . . .	52
WRegTest . . . . .	53
<b>Index</b>	<b>55</b>

---

 BJoint

*The Buckley-James censored regression estimator*


---

### Description

Compute the Buckley-James estimator in the regression model

$$y_i = \beta x_i + \epsilon_i$$

with right censored  $y_i$ . Iteration method.

### Usage

```
BJoint(x, y, delta, beta0 = NA, maxiter=30, error = 0.00001)
```

### Arguments

x	a matrix or vector containing the covariate, one row per observation.
y	a numeric vector of length N, censored responses.
delta	a vector of length N, delta=0/1 for censored/uncensored.
beta0	an optional vector for starting value of iteration.
maxiter	an optional integer to control iterations.
error	an optional positive value to control iterations.

## Details

This function compute the Buckley-James estimator when your model do not have an intercept term. Of course, if you include a column of 1's in the x matrix, it is also OK with this function and it is equivalent to having an intercept term. If your model do have an intercept term, then you could also (probably should) use the function `bj( )` in the `Design` library. It should be more refined than `BJoint` in the stopping rule for the iterations.

This function is included here mainly to produce the estimator value that may provide some useful information with the function `bjtest( )`. For example you may want to test a beta value near the Buckley-James estimator.

## Value

A list with the following components:

<code>beta</code>	the Buckley-James estimator.
<code>iteration</code>	number of iterations performed.

## Author(s)

Mai Zhou.

## References

Buckley, J. and James, I. (1979). Linear regression with censored data. *Biometrika*, **66** 429-36.

## Examples

```
x <- matrix(c(rnorm(50,mean=1), rnorm(50,mean=2)), ncol=2,nrow=50)
## Suppose now we wish to test Ho: 2mu(1)-mu(2)=0, then
y <- 2*x[,1]-x[,2]
xx <- c(28,-44,29,30,26,27,22,23,33,16,24,29,24,40,21,31,34,-2,25,19)
```

---

`bjtest`

*Test the Buckley-James estimator by Empirical Likelihood*

---

## Description

Use the empirical likelihood ratio and Wilks theorem to test if the regression coefficient is equal to beta.

The log empirical likelihood been maximized is

$$\sum_{d=1} \log \Delta F(e_i) + \sum_{d=0} \log[1 - F(e_i)];$$

where  $e_i$  are the residuals.

**Usage**

```
bjtest(y, d, x, beta)
```

**Arguments**

y a vector of length N, containing the censored responses.  
d a vector (length N) of either 1's or 0's. d=1 means y is uncensored; d=0 means y is right censored.  
x a matrix of size N by q.  
beta a vector of length q. The value of the regression coefficient to be tested in the model  $y_i = \beta x_i + \epsilon_i$

**Details**

The above likelihood should be understood as the likelihood of the error term, so in the regression model the error epsilon should be iid.

This version can handle the model where beta is a vector (of length q).

The estimation equations used when maximize the empirical likelihood is

$$0 = \sum d_i \Delta F(e_i)(x \cdot m[,i]) / (nw_i)$$

which was described in detail in the reference below.

**Value**

A list with the following components:

"-2LLR" the -2 loglikelihood ratio; have approximate chisq distribution under  $H_0$ .  
loge12 the log empirical likelihood, under estimating equation.  
loge1 the log empirical likelihood of the Kaplan-Meier of e's.  
prob the probabilities that max the empirical likelihood under estimating equation.

**Author(s)**

Mai Zhou.

**References**

Buckley, J. and James, I. (1979). Linear regression with censored data. *Biometrika*, **66** 429-36.  
Zhou, M. and Li, G. (2008). Empirical likelihood analysis of the Buckley-James estimator. *Journal of Multivariate Analysis* **99**, 649-664.  
Zhou, M. (2016) Empirical Likelihood Method in Survival Analysis. CRC Press.

**Examples**

```
xx <- c(28, -44, 29, 30, 26, 27, 22, 23, 33, 16, 24, 29, 24, 40, 21, 31, 34, -2, 25, 19)
```

bjtest1d

*Test the Buckley-James estimator by Empirical Likelihood, 1-dim only***Description**

Use the empirical likelihood ratio and Wilks theorem to test if the regression coefficient is equal to beta. For 1-dim beta only.

The log empirical likelihood been maximized is

$$\sum_{d=1} \log \Delta F(e_i) + \sum_{d=0} \log[1 - F(e_i)].$$

**Usage**

```
bjtest1d(y, d, x, beta)
```

**Arguments**

y	a vector of length N, containing the censored responses.
d	a vector of either 1's or 0's. d=1 means y is uncensored. d=0 means y is right censored.
x	a vector of length N, covariate.
beta	a number. the regression coefficient to be tested in the model $y = x \text{ beta} + \text{epsilon}$

**Details**

In the above likelihood,  $e_i = y_i - x * \text{beta}$  is the residuals.

Similar to `bjtest( )`, but only for 1-dim beta.

**Value**

A list with the following components:

"-2LLR"	the -2 loglikelihood ratio; have approximate chi square distribution under $H_0$ .
logel2	the log empirical likelihood, under estimating equation.
logel	the log empirical likelihood of the Kaplan-Meier of e's.
prob	the probabilities that max the empirical likelihood under estimating equation constraint.

**Author(s)**

Mai Zhou.

## References

- Buckley, J. and James, I. (1979). Linear regression with censored data. *Biometrika*, **66** 429-36.
- Owen, A. (1990). Empirical likelihood ratio confidence regions. *Ann. Statist.* **18** 90-120.
- Zhou, M. and Li, G. (2008). Empirical likelihood analysis of the Buckley-James estimator. *Journal of Multivariate Analysis*. 649-664.

## Examples

```
xx <- c(28, -44, 29, 30, 26, 27, 22, 23, 33, 16, 24, 29, 24, 40, 21, 31, 34, -2, 25, 19)
```

---

bjtestII	<i>Alternative test of the Buckley-James estimator by Empirical Likelihood</i>
----------	--------------------------------------------------------------------------------

---

## Description

Use the empirical likelihood ratio (alternative form) and Wilks theorem to test if the regression coefficient is equal to beta, based on the estimating equations.

The log empirical likelihood been maximized is

$$\sum_{j=1}^n \log p_j; \quad \sum p_j = 1$$

where the probability  $p_j$  is for the  $j$ th martingale differences of the estimating equations.

## Usage

```
bjtestII(y, d, x, beta)
```

## Arguments

- |      |                                                                                                                        |
|------|------------------------------------------------------------------------------------------------------------------------|
| y    | a vector of length N, containing the censored responses.                                                               |
| d    | a vector of length N. Either 1's or 0's. d=1 means y is uncensored; d=0 means y is right censored.                     |
| x    | a matrix of size N by q.                                                                                               |
| beta | a vector of length q. The value of the regression coefficient to be tested in the model $Y_i = \beta x_i + \epsilon_i$ |

## Details

The above likelihood should be understood as the likelihood of the martingale difference terms. For the definition of the Buckley-James martingale or estimating equation, please see the (2015) book in the reference list.

The estimation equations used when maximize the empirical likelihood is

$$0 = \sum d_i \Delta F(e_i)(x \cdot m[, i]) / (nw_i)$$

where  $e_i$  is the residuals, other details are described in the reference book of 2015 below.

The final test is carried out by `el.test`. So the output is similar to the output of `el.test`.

## Value

A list with the following components:

"-2LLR"	the -2 loglikelihood ratio; have approximate chisq distribution under $H_0$ .
logel2	the log empirical likelihood, under estimating equation.
logel	the log empirical likelihood of the Kaplan-Meier of e's.
prob	the probabilities that max the empirical likelihood under estimating equation.

## Author(s)

Mai Zhou.

## References

- Zhou, M. (2016) Empirical Likelihood Methods in Survival Analysis. CRC Press.
- Buckley, J. and James, I. (1979). Linear regression with censored data. *Biometrika*, **66** 429-36.
- Zhou, M. and Li, G. (2008). Empirical likelihood analysis of the Buckley-James estimator. *Journal of Multivariate Analysis*, **99**, 649–664.
- Zhu, H. (2007) Smoothed Empirical Likelihood for Quantiles and Some Variations/Extension of Empirical Likelihood for Buckley-James Estimator, Ph.D. dissertation, University of Kentucky.

## Examples

```
data(myeloma)
bjtestII(y=myeloma[,1], d=myeloma[,2], x=cbind(1, myeloma[,3]), beta=c(37, -3.4))
```

---

 el.cen.EM

*Empirical likelihood ratio for mean with right, left or doubly censored data, by EM algorithm*


---

### Description

This program uses EM algorithm to compute the maximized (wrt  $p_i$ ) empirical log likelihood function for right, left or doubly censored data with the MEAN constraint:

$$\sum_{d_i=1} p_i f(x_i) = \int f(t) dF(t) = \mu.$$

Where  $p_i = \Delta F(x_i)$  is a probability,  $d_i$  is the censoring indicator, 1(uncensored), 0(right censored), 2(left censored). It also returns those  $p_i$ .

The empirical log likelihood been maximized is

$$\sum_{d_i=1} \log \Delta F(x_i) + \sum_{d_i=0} \log[1 - F(x_i)] + \sum_{d_i=2} \log F(x_i).$$

### Usage

```
el.cen.EM(x,d,wt=rep(1,length(d)),fun=function(t){t},mu,maxit=25,error=1e-9,...)
```

### Arguments

x	a vector containing the observed survival times.
d	a vector containing the censoring indicators, 1-uncensored; 0-right censored; 2-left censored.
wt	a weight vector (case weight). positive. same length as d
fun	a left continuous (weight) function used to calculate the mean as in $H_0$ . fun(t) must be able to take a vector input t. Default to the identity function $f(t) = t$ .
mu	a real number used in the constraint, the mean value of $f(X)$ .
maxit	an optional integer, used to control maximum number of iterations.
error	an optional positive real number specifying the tolerance of iteration error. This is the bound of the $L_1$ norm of the difference of two successive weights.
...	additional arguments, if any, to pass to fun.

### Details

This implementation is all in R and have several for-loops in it. A faster version would use C to do the for-loop part. But this version seems faster enough and is easier to port to Splus.

We return the log likelihood all the time. Sometimes, (for right censored and no censor case) we also return the -2 log likelihood ratio. In other cases, you have to plot a curve with many values of the parameter, mu, to find out where is the place the log likelihood becomes maximum. And from



there you can get -2 log likelihood ratio between the maximum location and your current parameter in Ho.

In order to get a proper distribution as NPMLE, we automatically change the  $d$  for the largest observation to 1 (even if it is right censored), similar for the left censored, smallest observation.  $\mu$  is a given constant. When the given constants  $\mu$  is too far away from the NPMLE, there will be no distribution satisfy the constraint. In this case the computation will stop. The -2 Log empirical likelihood ratio should be infinite.

The constant mu must be inside  $(\min f(x_i), \max f(x_i))$  for the computation to continue. It is always true that the NPMLE values are feasible. So when the computation stops, try move the mu closer to the NPMLE —

$$\sum_{d_i=1} p_i^0 f(x_i)$$

$p_i^0$  taken to be the jumps of the NPMLE of CDF. Or use a different fun.

Difference to the function e1.cen.EM2: here duplicate (input) observations are collapsed (with weight 2, 3, ... etc.) but those will stay separate by default in the e1.cen.EM2. This will lead to a different loglik value. But the -2LLR value should be same in either version.

### Value

A list with the following components:

loglik	the maximized empirical log likelihood under the constraint.
times	locations of CDF that have positive mass.
prob	the jump size of CDF at those locations.
"-2LLR"	If available, it is Minus two times the Empirical Log Likelihood Ratio. Should be approximately chi-square distributed under Ho.
Pval	The P-value of the test, using chi-square approximation.
lam	The Lagrange multiplier. Added 5/2007.

### Author(s)

Mai Zhou

### References

Zhou, M. (2005). Empirical likelihood ratio with arbitrary censored/truncated data by EM algorithm. *Journal of Computational and Graphical Statistics*, 643-656.

Murphy, S. and van der Vaart (1997) Semiparametric likelihood ratio inference. *Ann. Statist.* **25**, 1471-1509.

### Examples

```
## example with tied observations
x <- c(1, 1.5, 2, 3, 4, 5, 6, 5, 4, 1, 2, 4.5)
d <- c(1, 1, 0, 1, 0, 1, 1, 1, 1, 0, 0, 1)
e1.cen.EM(x,d,mu=3.5)
## we should get "-2LLR" = 1.2466...
```

```

myfun5 <- function(x, theta, eps) {
  u <- (x-theta)*sqrt(5)/eps
  INDE <- (u < sqrt(5)) & (u > -sqrt(5))
  u[u >= sqrt(5)] <- 0
  u[u <= -sqrt(5)] <- 1
  y <- 0.5 - (u - (u)^3/15)*3/(4*sqrt(5))
  u[ INDE ] <- y[ INDE ]
  return(u)
}
e1.cen.EM(x, d, fun=myfun5, mu=0.5, theta=3.5, eps=0.1)
## example of using wt in the input. Since the x-vector contain
## two 5 (both d=1), and two 2(both d=0), we can also do
xx <- c(1, 1.5, 2, 3, 4, 5, 6, 4, 1, 4.5)
dd <- c(1, 1, 0, 1, 0, 1, 1, 1, 0, 1)
wt <- c(1, 1, 2, 1, 1, 2, 1, 1, 1, 1)
e1.cen.EM(x=xx, d=dd, wt=wt, mu=3.5)
## this should be the same as the first example.

```

---

e1.cen.EM2

*Empirical likelihood ratio test for a vector of means with right, left or doubly censored data, by EM algorithm*


---

### Description

This function is similar to `e1.cen.EM()`, but for multiple constraints. In the input there is a vector of observations  $x = (x_1, \dots, x_n)$  and a function `fun`. The function `fun` should return the (n by k) matrix

$$(f_1(x), f_2(x), \dots, f_k(x)).$$

Also, the ordering of the observations, when consider censoring or redistributing-to-the-right, is according to the value of  $x$ , not  $\text{fun}(x)$ . So the probability distribution is for values  $x$ . This program uses EM algorithm to maximize (wrt  $p_i$ ) empirical log likelihood function for right, left or doubly censored data with the MEAN constraint:

$$j = 1, 2, \dots, k \quad \sum_{d_i=1} p_i f_j(x_i) = \int f_j(t) dF(t) = \mu_j.$$

Where  $p_i = \Delta F(x_i)$  is a probability,  $d_i$  is the censoring indicator, 1(uncensored), 0(right censored), 2(left censored). It also returns those  $p_i$ . The log likelihood function is defined as

$$\sum_{d_i=1} \log \Delta F(x_i) + \sum_{d_i=2} \log F(x_i) + \sum_{d_i=0} \log[1 - F(x_i)].$$

### Usage

```
e1.cen.EM2(x,d,xc=1:length(x),fun,mu,maxit=25,error=1e-9,...)
```

**Arguments**

x	a vector containing the observed survival times.
d	a vector containing the censoring indicators, 1-uncensored; 0-right censored; 2-left censored.
xc	an optional vector of collapsing control values. If xc[i] xc[j] have different values then (x[i], d[i]), (x[j], d[j]) will not merge into one observation with weight two, even if they are identical. Default is not to merge.
fun	a left continuous (weight) function that returns a matrix. The columns (=k) of the matrix is used to calculate the means and will be tested in $H_0$ . fun(t) must be able to take a vector input t.
mu	a vector of length k. Used in the constraint, as the mean of $f(X)$ .
maxit	an optional integer, used to control maximum number of iterations.
error	an optional positive real number specifying the tolerance of iteration error. This is the bound of the $L_1$ norm of the difference of two successive weights.
...	additional inputs to pass to fun().

**Details**

This implementation is all in R and have several for-loops in it. A faster version would use C to do the for-loop part. (but this version is easier to port to Splus, and seems faster enough).

We return the log likelihood all the time. Sometimes, (for right censored and no censor case) we also return the -2 log likelihood ratio. In other cases, you have to plot a curve with many values of the parameter, mu, to find out where the log likelihood becomes maximum. And from there you can get -2 log likelihood ratio between the maximum location and your current parameter in  $H_0$ .

In order to get a proper distribution as NPMLE, we automatically change the  $d$  for the largest observation to 1 (even if it is right censored), similar for the left censored, smallest observation.  $\mu$  is a given constant vector. When the given constants  $\mu$  is too far away from the NPMLE, there will be no distribution satisfy the constraint. In this case the computation will stop. The -2 Log empirical likelihood ratio should be infinite.

The constant vector mu must be inside  $(\min f(x_i), \max f(x_i))$  for the computation to continue. It is always true that the NPMLE values are feasible. So when the computation stops, try move the mu closer to the NPMLE —

$$\hat{\mu}_j = \sum_{d_i=1} p_i^0 f_j(x_i)$$

where  $p_i^0$  taken to be the jumps of the NPMLE of CDF. Or use a different fun.

Difference to the function el.cen.EM: due to the introduction of input xc here in this function, the output loglik may be different compared to the function el.cen.EM due to not collapsing of duplicated input survival values. The -2LLR should be the same from both functions.

**Value**

A list with the following components:

loglik	the maximized empirical log likelihood under the constraints.
times	locations of CDF that have positive mass.

prob	the jump size of CDF at those locations.
"-2LLR"	If available, it is Minus two times the Empirical Log Likelihood Ratio. Should be approx. chi-square distributed under Ho.
Pval	If available, the P-value of the test, using chi-square approximation.
lam	the Lagrange multiplier in the final EM step. (the M-step)

### Author(s)

Mai Zhou

### References

- Zhou, M. (2005). Empirical likelihood ratio with arbitrary censored/truncated data by EM algorithm. *Journal of Computational and Graphical Statistics*, 643-656.
- Zhou, M. (2002). Computing censored empirical likelihood ratio by EM algorithm. *Tech Report, Univ. of Kentucky, Dept of Statistics*

### Examples

```
## censored regression with one right censored observation.
## we check the estimation equation, with the MLE inside myfun7.
y <- c(3, 5.3, 6.4, 9.1, 14.1, 15.4, 18.1, 15.3, 14, 5.8, 7.3, 14.4)
x <- c(1, 1.5, 2, 3, 4, 5, 6, 5, 4, 1, 2, 4.5)
d <- c(1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 0)
### first we estimate beta, the MLE
lm.wfit(x=cbind(rep(1,12),x), y=y, w=WKM(x=y, d=d)$jump[rank(y)])$coef
## you should get 1.392885 and 2.845658
## then define myfun7 with the MLE value
myfun7 <- function(y, xmat) {
temp1 <- y - ( 1.392885 + 2.845658 * xmat)
return( cbind( temp1, xmat*temp1) )
}
## now test
el.cen.EM2(y,d, fun=myfun7, mu=c(0,0), xmat=x)
## we should get, Pval = 1 , as the MLE should.
## for other values of (a, b) inside myfun7, you get other Pval
##
rqfun1 <- function(y, xmat, beta, tau = 0.5) {
temp1 <- tau - (1-myfun55(y-beta*xmat))
return(xmat * temp1)
}
myfun55 <- function(x, eps=0.001){
u <- x*sqrt(5)/eps
INDE <- (u < sqrt(5)) & (u > -sqrt(5))
u[u >= sqrt(5)] <- 0
u[u <= -sqrt(5)] <- 1
y <- 0.5 - (u - (u)^3/15)*3/(4*sqrt(5))
u[ INDE ] <- y[ INDE ]
return(u)
}
## myfun55 is a smoothed indicator fn.
```

```

## eps should be between (1/sqrt(n), 1/n^0.75) [Chen and Hall]
el.cen.EM2(x=y,d=d,xc=1:12,fun=rqfun1,mu=0,xmat=x,beta=3.08,tau=0.44769875)
## default tau=0.5
el.cen.EM2(x=y,d=d,xc=1:12,fun=rqfun1,mu=0,xmat=x,beta=3.0799107404)
#####
### next 2 examples are testing the mean/median residual time
#####
mygfun <- function(s, age, muage) {as.numeric(s >= age)*(s-(age+muage))}
mygfun2 <- function(s, age, Mdage)
  {as.numeric(s <= (age+Mdage)) - 0.5*as.numeric(s <= age)}
## Not run:
library(survival)
time <- cancer$time
status <- cancer$status-1
###for mean residual time
el.cen.EM2(x=time, d=status, fun=mygfun, mu=0, age=365.25, muage=234)$Pval
el.cen.EM2(x=time, d=status, fun=mygfun, mu=0, age=365.25, muage=323)$Pval
### for median residual time
el.cen.EM2(x=time, d=status, fun=mygfun2, mu=0.5, age=365.25, Mdage=184)$Pval
el.cen.EM2(x=time, d=status, fun=mygfun2, mu=0.5, age=365.25, Mdage=321)$Pval

## End(Not run)
## Not run:
#### For right censor only data (Kaplan-Meier) we can use this function to get a faster computation
#### by calling the kmc 0.2-2 package.
el.cen.R <- function (x, d, xc = 1:length(x), fun, mu, error = 1e-09, ...)
{
  xvec <- as.vector(x)
  d <- as.vector(d)
  mu <- as.vector(mu)
  xc <- as.vector(xc)
  n <- length(d)
  if (length(xvec) != n)
    stop("length of d and x must agree")
  if (length(xc) != n)
    stop("length of xc and d must agree")
  if (n <= 2 * length(mu) + 1)
    stop("Need more observations")
  if (any((d != 0) & (d != 1) ))
    stop("d must be 0(right-censored) or 1(uncensored)")
  if (!is.numeric(xvec))
    stop("x must be numeric")
  if (!is.numeric(mu))
    stop("mu must be numeric")

  funx <- as.matrix(fun(xvec, ...))
  pp <- ncol(funx)
  if (length(mu) != pp)
    stop("length of mu and ncol of fun(x) must agree")
  temp <- Wdataclean5(z = xvec, d, zc = xc, xmat = funx)
  x <- temp$value
  d <- temp$d
  w <- temp$weight

```

```

funx <- temp$xxmat
d[length(d)] <- 1
xd1 <- x[d == 1]
if (length(xd1) <= 1)
stop("need more distinct uncensored obs.")
funxd1 <- funx[d == 1, ]
xd0 <- x[d == 0]
wd1 <- w[d == 1]
wd0 <- w[d == 0]
m <- length(xd0)

pnew <- NA
num <- NA
if (m > 0) {
gfun <- function(x) { return( fun(x, ...) - mu ) }
temp <- kmc.solve(x=x, d=d, g=list(gfun))
logel <- temp$loglik.h0
logel00 <- temp$loglik.null
lam <- - temp$lambada
}
if (m == 0) {
num <- 0
temp6 <- el.test.wt2(x = funxd1, wt = wd1, mu)
pnew <- temp6$prob
lam <- temp6$lambada
logel <- sum(wd1 * log(pnew))
logel00 <- sum(wd1 * log(wd1/sum(wd1)))
}
tval <- 2 * (logel00 - logel)
list(loglik = logel, times = xd1, prob = pnew, lam = lam,
iters = num, `-2LLR` = tval, Pval = 1 - pchisq(tval,
df = length(mu)))
}

## End(Not run)

```

---

el.cen.test

*Empirical likelihood ratio for mean with right censored data, by QP.*


---

### Description

This program computes the maximized (wrt  $p_i$ ) empirical log likelihood function for right censored data with the MEAN constraint:

$$\sum_i [d_i p_i g(x_i)] = \int g(t) dF(t) = \mu$$

where  $p_i = \Delta F(x_i)$  is a probability,  $d_i$  is the censoring indicator. The  $d$  for the largest observation is always taken to be 1. It then computes the -2 log empirical likelihood ratio which should be

approximately chi-square distributed if the constraint is true. Here  $F(t)$  is the (unknown) CDF;  $g(t)$  can be any given left continuous function in  $t$ .  $\mu$  is a given constant. The data must contain some right censored observations. If there is no censoring or the only censoring is the largest observation, the code will stop and we should use `el.test()` which is for uncensored data.

The log empirical likelihood been maximized is

$$\sum_{d_i=1} \log \Delta F(x_i) + \sum_{d_i=0} \log[1 - F(x_i)].$$

### Usage

```
el.cen.test(x,d,fun=function(x){x},mu,error=1e-8,maxit=15)
```

### Arguments

<code>x</code>	a vector containing the observed survival times.
<code>d</code>	a vector containing the censoring indicators, 1-uncensor; 0-censor.
<code>fun</code>	a left continuous (weight) function used to calculate the mean as in $H_0$ . <code>fun(t)</code> must be able to take a vector input <code>t</code> . Default to the identity function $f(t) = t$ .
<code>mu</code>	a real number used in the constraint, sum to this value.
<code>error</code>	an optional positive real number specifying the tolerance of iteration error in the QP. This is the bound of the $L_1$ norm of the difference of two successive weights.
<code>maxit</code>	an optional integer, used to control maximum number of iterations.

### Details

When the given constants  $\mu$  is too far away from the NPMLE, there will be no distribution satisfy the constraint. In this case the computation will stop. The -2 Log empirical likelihood ratio should be infinite.

The constant `mu` must be inside  $(\min f(x_i), \max f(x_i))$  for the computation to continue. It is always true that the NPMLE values are feasible. So when the computation cannot continue, try move the `mu` closer to the NPMLE, or use a different `fun`.

This function depends on `Wdataclean2()`, `WKM()` and `solve3.QP()`

This function uses sequential Quadratic Programming to find the maximum. Unlike other functions in this package, it can be slow for larger sample sizes. It took about one minute for a sample of size 2000 with 20% censoring on a 1GHz, 256MB PC, about 19 seconds on a 3 GHz 512MB PC.

### Value

A list with the following components:

<code>"-2LLR"</code>	The -2Log Likelihood ratio.
<code>xtimes</code>	the location of the CDF jumps.
<code>weights</code>	the jump size of CDF at those locations.
<code>Pval</code>	P-value
<code>error</code>	the $L_1$ norm between the last two wts.
<code>iteration</code>	number of iterations carried out

**Author(s)**

Mai Zhou, Kun Chen

**References**

Pan, X. and Zhou, M. (1999). Using 1-parameter sub-family of distributions in empirical likelihood ratio with censored data. *J. Statist. Plann. Inference*. **75**, 379-392.

Chen, K. and Zhou, M. (2000). Computing censored empirical likelihood ratio using Quadratic Programming. *Tech Report, Univ. of Kentucky, Dept of Statistics*

Zhou, M. and Chen, K. (2007). Computation of the empirical likelihood ratio from censored data. *Journal of Statistical Computing and Simulation*, **77**, 1033-1042.

**Examples**

```
e1.cen.test(rexp(100), c(rep(0,25),rep(1,75)), mu=1.5)
## second example with tied observations
x <- c(1, 1.5, 2, 3, 4, 5, 6, 5, 4, 1, 2, 4.5)
d <- c(1, 1, 0, 1, 0, 1, 1, 1, 1, 0, 0, 1)
e1.cen.test(x,d,mu=3.5)
# we should get "-2LLR" = 1.246634 etc.
```

---

e1.ltrc.EM

---

*Empirical likelihood ratio for mean with left truncated and right censored data, by EM algorithm*


---

**Description**

This program uses EM algorithm to compute the maximized (wrt  $p_i$ ) empirical log likelihood function for left truncated and right censored data with the MEAN constraint:

$$\sum_{d_i=1} p_i f(x_i) = \int f(t) dF(t) = \mu .$$

Where  $p_i = \Delta F(x_i)$  is a probability,  $d_i$  is the censoring indicator, 1(uncensored), 0(right censored). The  $d$  for the largest observation  $x$ , is always (automatically) changed to 1.  $\mu$  is a given constant. This function also returns those  $p_i$ .

The log empirical likelihood function been maximized is

$$\sum_{d_i=1} \log \frac{\Delta F(x_i)}{1 - F(y_i)} + \sum_{d_i=0} \log \frac{1 - F(x_i)}{1 - F(y_i)} .$$

**Usage**

```
e1.ltrc.EM(y,x,d,fun=function(t){t},mu,maxit=30,error=1e-9)
```



**Arguments**

y	an optional vector containing the observed left truncation times.
x	a vector containing the censored survival times.
d	a vector containing the censoring indicators, 1-uncensored; 0-right censored.
fun	a continuous (weight) function used to calculate the mean as in $H_0$ . $\text{fun}(t)$ must be able to take a vector input $t$ . Default to the identity function $f(t) = t$ .
mu	a real number used in the constraint, mean value of $f(X)$ .
error	an optional positive real number specifying the tolerance of iteration error. This is the bound of the $L_1$ norm of the difference of two successive weights.
maxit	an optional integer, used to control maximum number of iterations.

**Details**

We return the -2 log likelihood ratio, and the constrained NPMLE of CDF. The un-constrained NPMLE should be WJT or Lynden-Bell estimator.

When the given constants  $\mu$  is too far away from the NPMLE, there will be no distribution satisfy the constraint. In this case the computation will stop. The -2 Log empirical likelihood ratio should be infinite.

The constant mu must be inside  $(\min f(x_i), \max f(x_i))$  for the computation to continue. It is always true that the NPMLE values are feasible. So when the computation stops, try move the mu closer to the NPMLE —

$$\sum_{d_i=1} p_i^0 f(x_i)$$

$p_i^0$  taken to be the jumps of the NPMLE of CDF. Or use a different fun.

This implementation is all in R and have several for-loops in it. A faster version would use C to do the for-loop part. (but this version is easier to port to Splus, and seems faster enough).

**Value**

A list with the following components:

times	locations of CDF that have positive mass.
prob	the probability of the constrained NPMLE of CDF at those locations.
"-2LLR"	It is Minus two times the Empirical Log Likelihood Ratio. Should be approximate chi-square distributed under $H_0$ .

**Author(s)**

Mai Zhou

## References

- Zhou, M. (2002). Computing censored and truncated empirical likelihood ratio by EM algorithm. *Tech Report, Univ. of Kentucky, Dept of Statistics*
- Tsai, W. Y., Jewell, N. P., and Wang, M. C. (1987). A note on product-limit estimator under right censoring and left truncation. *Biometrika*, **74**, 883-886.
- Turnbull, B. (1976). The empirical distribution function with arbitrarily grouped, censored and truncated data. *JRSS B*, 290-295.
- Zhou, M. (2005). Empirical likelihood ratio with arbitrarily censored/truncated data by EM algorithm. *Journal of Computational and Graphical Statistics* **14**, 643-656.

## Examples

```
## example with tied observations
y <- c(0, 0, 0.5, 0, 1, 2, 2, 0, 0, 0, 0, 0)
x <- c(1, 1.5, 2, 3, 4, 5, 6, 5, 4, 1, 2, 4.5)
d <- c(1, 1, 0, 1, 0, 1, 1, 1, 1, 0, 0, 1)
el.ltrc.EM(y,x,d,mu=3.5)
ypsy <- c(51, 58, 55, 28, 25, 48, 47, 25, 31, 30, 33, 43, 45, 35, 36)
xpsy <- c(52, 59, 57, 50, 57, 59, 61, 61, 62, 67, 68, 69, 69, 65, 76)
dpsy <- c(1, 1, 1, 1, 1, 1, 1, 1, 0, 0, 0, 1, 1, 0, 1)
el.ltrc.EM(ypsy,xpsy,dpsy,mu=64)
```

---

el.test

*Empirical likelihood ratio test for the means, uncensored data*

---

## Description

Compute the empirical likelihood ratio with the mean vector fixed at mu.

The log empirical likelihood been maximized is

$$\sum_{i=1}^n \log \Delta F(x_i).$$

## Usage

```
el.test(x, mu, lam, maxit=25, gradtol=1e-7,
        svdtol = 1e-9, itertrace=FALSE)
```

## Arguments

- x** a matrix or vector containing the data, one row per observation.
- mu** a numeric vector (of length = ncol(x)) to be tested as the mean vector of x above, as  $H_0$ .
- lam** an optional vector of length = length(mu), the starting value of Lagrange multipliers, will use 0 if missing.

maxit	an optional integer to control iteration when solve constrained maximization.
gradtol	an optional real value for convergence test.
svdtol	an optional real value to detect singularity while solve equations.
itertrace	a logical value. If the iteration history needs to be printed out.

### Details

If  $\mu$  is in the interior of the convex hull of the observations  $x$ , then  $wts$  should sum to  $n$ . If  $\mu$  is outside the convex hull then  $wts$  should sum to nearly zero, and  $-2LLR$  will be a large positive number. It should be infinity, but for inferential purposes a very large number is essentially equivalent. If  $\mu$  is on the boundary of the convex hull then  $wts$  should sum to nearly  $k$  where  $k$  is the number of observations within that face of the convex hull which contains  $\mu$ .

When  $\mu$  is interior to the convex hull, it is typical for the algorithm to converge quadratically to the solution, perhaps after a few iterations of searching to get near the solution. When  $\mu$  is outside or near the boundary of the convex hull, then the solution involves a lambda of infinite norm. The algorithm tends to nearly double lambda at each iteration and the gradient size then decreases roughly by half at each iteration.

The goal in writing the algorithm was to have it “fail gracefully” when  $\mu$  is not inside the convex hull. The user can either leave  $-2LLR$  “large and positive” or can replace it by infinity when the weights do not sum to nearly  $n$ .

### Value

A list with the following components:

$-2LLR$	the $-2$ loglikelihood ratio; approximate chisq distribution under $H_0$ .
Pval	the observed P-value by chi-square approximation.
lambda	the final value of Lagrange multiplier.
grad	the gradient at the maximum.
hess	the Hessian matrix.
wts	weights on the observations
nits	number of iteration performed

### Author(s)

Original Splus code by Art Owen. Adapted to R by Mai Zhou.

### References

Owen, A. (1990). Empirical likelihood ratio confidence regions. *Ann. Statist.* **18**, 90-120.

**Examples**

```
x <- matrix(c(rnorm(50,mean=1), rnorm(50,mean=2)), ncol=2,nrow=50)
el.test(x, mu=c(1,2))
## Suppose now we wish to test Ho: 2mu(1)-mu(2)=0, then
y <- 2*x[,1]-x[,2]
el.test(y, mu=0)
xx <- c(28,-44,29,30,26,27,22,23,33,16,24,29,24,40,21,31,34,-2,25,19)
el.test(xx, mu=15) ##### -2LLR = 1.805702
```

---

el.test.wt

*Weighted Empirical Likelihood ratio for mean, uncensored data*


---

**Description**

This program is similar to `el.test( )` except it takes weights, and is for one dimensional  $\mu$ .

The mean constraint considered is:

$$\sum_{i=1}^n p_i x_i = \mu.$$

where  $p_i = \Delta F(x_i)$  is a probability. Plus the probability constraint:  $\sum p_i = 1$ .

The weighted log empirical likelihood been maximized is

$$\sum_{i=1}^n w_i \log p_i.$$

**Usage**

```
el.test.wt(x, wt, mu, usingC=TRUE)
```

**Arguments**

<code>x</code>	a vector containing the observations.
<code>wt</code>	a vector containing the weights.
<code>mu</code>	a real number used in the constraint, weighted mean value of $f(X)$ .
<code>usingC</code>	TRUE: use C function, which may be benefit when sample size is large; FALSE: use pure R function.

**Details**

This function used to be an internal function. It becomes external because others may find it useful elsewhere.

The constant  $\mu$  must be inside  $(\min x_i, \max x_i)$  for the computation to continue.

**Value**

A list with the following components:

x	the observations.
wt	the vector of weights.
prob	The probabilities that maximized the weighted empirical likelihood under mean constraint.

**Author(s)**

Mai Zhou, Y.F. Yang for C part.

**References**

- Owen, A. (1990). Empirical likelihood ratio confidence regions. *Ann. Statist.* **18**, 90-120.
- Zhou, M. (2002). Computing censored empirical likelihood ratio by EM algorithm. *Tech Report, Univ. of Kentucky, Dept of Statistics*

**Examples**

```
## example with tied observations
x <- c(1, 1.5, 2, 3, 4, 5, 6, 5, 4, 1, 2, 4.5)
d <- c(1, 1, 0, 1, 0, 1, 1, 1, 1, 0, 0, 1)
el.cen.EM(x,d,mu=3.5)
## we should get "-2LLR" = 1.2466...
myfun5 <- function(x, theta, eps) {
  u <- (x-theta)*sqrt(5)/eps
  INDE <- (u < sqrt(5)) & (u > -sqrt(5))
  u[u >= sqrt(5)] <- 0
  u[u <= -sqrt(5)] <- 1
  y <- 0.5 - (u - (u^3/15)*3)/(4*sqrt(5))
  u[ INDE ] <- y[ INDE ]
  return(u)
}
el.cen.EM(x, d, fun=myfun5, mu=0.5, theta=3.5, eps=0.1)
```

---

el.test.wt2

*Weighted Empirical Likelihood ratio for mean(s), uncensored data*


---

**Description**

This program is similar to `el.test()` except it takes weights.

The mean constraints are:

$$\sum_{i=1}^n p_i x_i = \mu.$$

Where  $p_i = \Delta F(x_i)$  is a probability. Plus the probability constraint:  $\sum p_i = 1$ .

The weighted log empirical likelihood been maximized is

$$\sum_{i=1}^n w_i \log p_i.$$

### Usage

```
el.test.wt2(x, wt, mu, maxit = 25, gradtol = 1e-07, Hessian = FALSE,
            svdtol = 1e-09, itertrace = FALSE)
```

### Arguments

x	a matrix (of size nxp) or vector containing the observations.
wt	a vector of length n, containing the weights. If weights are all 1, this is very simila to el.test. wt have to be positive.
mu	a vector of length p, used in the constraint. weighted mean value of $f(X)$ .
maxit	an integer, the maximum number of iteration.
gradtol	a positive real number, the tolerance for a solution
Hessian	logical. if the Hessian needs to be computed?
svdtol	tolerance in perform SVD of the Hessian matrix.
itertrace	TRUE/FALSE, if the intermediate steps needs to be printed.

### Details

This function used to be an internal function. It becomes external because others may find it useful.

It is similar to the function `el.test()` with the following differences:

- (1) The output lambda in `el.test.wts`, when divided by n (the sample size or sum of all the weights) should be equal to the output lambda in `el.test`.
- (2) The Newton step of iteration in `el.test.wts` is different from those in `el.test`. (even when all the weights are one).

### Value

A list with the following components:

lambda	the Lagrange multiplier. Solution.
wt	the vector of weights.
grad	The gradian at the final solution.
nits	number of iterations performed.
prob	The probabilities that maximized the weighted empirical likelihood under mean constraint.

### Author(s)

Mai Zhou

## References

- Owen, A. (1990). Empirical likelihood ratio confidence regions. *Ann. Statist.* **18**, 90-120.
- Zhou, M. (2002). Computing censored empirical likelihood ratio by EM algorithm. *Tech Report, Univ. of Kentucky, Dept of Statistics*

## Examples

```
## example with tied observations
x <- c(1, 1.5, 2, 3, 4, 5, 6, 5, 4, 1, 2, 4.5)
d <- c(1, 1, 0, 1, 0, 1, 1, 1, 1, 0, 0, 1)
el.cen.EM(x,d,mu=3.5)
## we should get "-2LLR" = 1.2466...
myfun5 <- function(x, theta, eps) {
  u <- (x-theta)*sqrt(5)/eps
  INDE <- (u < sqrt(5)) & (u > -sqrt(5))
  u[u >= sqrt(5)] <- 0
  u[u <= -sqrt(5)] <- 1
  y <- 0.5 - (u - (u^3/15)*3)/(4*sqrt(5))
  u[ INDE ] <- y[ INDE ]
  return(u)
}
el.cen.EM(x, d, fun=myfun5, mu=0.5, theta=3.5, eps=0.1)
```

---

el.trun.test

*Empirical likelihood ratio for mean with left truncated data*

---

## Description

This program uses EM algorithm to compute the maximized (wrt  $p_i$ ) empirical log likelihood function for left truncated data with the MEAN constraint:

$$\sum p_i f(x_i) = \int f(t) dF(t) = \mu .$$

Where  $p_i = \Delta F(x_i)$  is a probability.  $\mu$  is a given constant. It also returns those  $p_i$  and the  $p_i$  without constraint, the Lynden-Bell estimator.

The log likelihood been maximized is

$$\sum_{i=1}^n \log \frac{\Delta F(x_i)}{1 - F(y_i)} .$$

## Usage

```
el.trun.test(y,x,fun=function(t){t},mu,maxit=20,error=1e-9)
```

**Arguments**

y	a vector containing the left truncation times.
x	a vector containing the survival times. truncation means $x > y$ .
fun	a continuous (weight) function used to calculate the mean as in $H_0$ . fun(t) must be able to take a vector input t. Default to the identity function $f(t) = t$ .
mu	a real number used in the constraint, mean value of $f(X)$ .
error	an optional positive real number specifying the tolerance of iteration error. This is the bound of the $L_1$ norm of the difference of two successive weights.
maxit	an optional integer, used to control maximum number of iterations.

**Details**

This implementation is all in R and have several for-loops in it. A faster version would use C to do the for-loop part. But it seems faster enough and is easier to port to Splus.

When the given constants  $\mu$  is too far away from the NPMLE, there will be no distribution satisfy the constraint. In this case the computation will stop. The -2 Log empirical likelihood ratio should be infinite.

The constant mu must be inside  $(\min f(x_i), \max f(x_i))$  for the computation to continue. It is always true that the NPMLE values are feasible. So when the computation stops, try move the mu closer to the NPMLE —

$$\sum_{d_i=1} p_i^0 f(x_i)$$

$p_i^0$  taken to be the jumps of the NPMLE of CDF. Or use a different fun.

**Value**

A list with the following components:

"-2LLR"	the maximized empirical log likelihood ratio under the constraint.
NPMLE	jumps of NPMLE of CDF at ordered x.
NPMLEmu	same jumps but for constrained NPMLE.

**Author(s)**

Mai Zhou

**References**

- Zhou, M. (2005). Empirical likelihood ratio with arbitrary censored/truncated data by EM algorithm. *Journal of Computational and Graphical Statistics*, **14**, 643-656.
- Li, G. (1995). Nonparametric likelihood ratio estimation of probabilities for truncated data. *JASA* **90**, 997-1003.
- Turnbull (1976). The empirical distribution function with arbitrarily grouped, censored and truncated data. *JRSS B* **38**, 290-295.





## Details

The log likelihood been maximized is the ‘binomial’ empirical likelihood:

$$\sum D_i \log w_i + (R_i - D_i) \log[1 - w_i]$$

where  $w_i = \Delta H(t_i)$  is the jump of the cumulative hazard function,  $D_i$  is the number of failures observed at  $t_i$ ,  $R_i$  is the number of subjects at risk at time  $t_i$ .

For discrete distributions, the jump size of the cumulative hazard at the last jump is always 1. We have to exclude this jump from the summation since  $\log(1 - dH(\cdot))$  do not make sense.

The constants theta and K must be inside the so called feasible region for the computation to continue. This is similar to the requirement that in testing the value of the mean, the value must be inside the convex hull of the observations. It is always true that the NPMLE values are feasible. So when the computation stops, try move the theta and K closer to the NPMLE. When the computation stops, the -2LLR should have value infinite.

In case you do not need the theta in the definition of the function  $f$ , you still need to formally define your fun function with a theta input, just to match the arguments.

## Value

A list with the following components:

times	the location of the hazard jumps.
wts	the jump size of hazard function at those locations.
lambda	the final value of the Lagrange multiplier.
"-2LLR"	The discrete -2Log Likelihood ratio.
Pval	P-value
niters	number of iterations used

## Author(s)

Mai Zhou

## References

- Fang, H. (2000). Binomial Empirical Likelihood Ratio Method in Survival Analysis. Ph.D. Thesis, Univ. of Kentucky, Dept of Statistics.
- Zhou and Fang (2001). “Empirical likelihood ratio for 2 sample problem for censored data”. *Tech Report, Univ. of Kentucky, Dept of Statistics*
- Zhou, M. and Fang, H. (2006). A comparison of Poisson and binomial empirical likelihood. *Tech Report, Univ. of Kentucky, Dept of Statistics*

**Examples**

```

fun4 <- function(x, theta) { as.numeric(x <= theta) }
x <- c(1, 2, 3, 4, 5, 6, 5, 4, 3, 4, 1, 2.4, 4.5)
d <- c(1, 0, 1, 0, 1, 0, 1, 0, 1, 1, 0, 1, 1)
# test if -H(4) = -0.7
emplikH.disc(x=x,d=d,K=-0.7,fun=fun4,theta=4)
# we should get "-2LLR" 0.1446316 etc....
y <- c(-2,-2, -2, 1.5, -1)
emplikH.disc(x=x,d=d,y=y,K=-0.7,fun=fun4,theta=4)

```

emplikH.disc2

*Two sample empirical likelihood ratio for discrete hazards with right censored, left truncated data, one parameter.*

**Description**

Use empirical likelihood ratio and Wilks theorem to test the null hypothesis that

$$\int f_1(t)I_{[dH_1 < 1]} \log(1 - dH_1(t)) - \int f_2(t)I_{[dH_2 < 1]} \log(1 - dH_2(t)) = \theta$$

where  $H_*(t)$  is the (unknown) discrete cumulative hazard function;  $f_*(t)$  can be any predictable functions of  $t$ .  $\theta$  is the parameter. The given value of  $\theta$  in these computation are the value to be tested. The data can be right censored and left truncated.

When the given constants  $\theta$  is too far away from the NPMLE, there will be no hazard function satisfy this constraint and the -2 Log empirical likelihood ratio will be infinite. In this case the computation will stop.

**Usage**

```

emplikH.disc2(x1, d1, y1= -Inf, x2, d2, y2 = -Inf,
              theta, fun1, fun2, tola = 1e-6, maxi, mini)

```

**Arguments**

x1	a vector, the observed survival times, sample 1.
d1	a vector, the censoring indicators, 1-uncensor; 0-censor.
y1	optional vector, the left truncation times.
x2	a vector, the observed survival times, sample 2.
d2	a vector, the censoring indicators, 1-uncensor; 0-censor.
y2	optional vector, the left truncation times.
fun1	a predictable function used to calculate the weighted discrete hazard in $H_0$ . fun1(x) must be able to take a vector input x.
fun2	similar to fun1, but for sample 2.

tola	an optional positive real number, the tolerance of iteration error in solve the non-linear equation needed in constrained maximization.
theta	a given real number. for Ho constraint.
maxi	upper bound for lambda, usually positive.
mini	lower bound for lambda, usually negative.

### Details

The log likelihood been maximized is the ‘binomial’ empirical likelihood:

$$\sum D_{1i} \log w_i + (R_{1i} - D_{1i}) \log[1 - w_i] + \sum D_{2j} \log v_j + (R_{2j} - D_{2j}) \log[1 - v_j]$$

where  $w_i = \Delta H_1(t_i)$  is the jump of the cumulative hazard function at  $t_i$ ,  $D_{1i}$  is the number of failures observed at  $t_i$ ,  $R_{1i}$  is the number of subjects at risk at time  $t_i$ .

For discrete distributions, the jump size of the cumulative hazard at the last jump is always 1. We have to exclude this jump from the summation in the constraint calculation since  $\log(1 - dH(\cdot))$  do not make sense.

The constants theta must be inside the so called feasible region for the computation to continue. This is similar to the requirement that in ELR testing the value of the mean, the value must be inside the convex hull of the observations. It is always true that the NPMLE values are feasible. So when the computation stops, try move the theta closer to the NPMLE. When the computation stops, the -2LLR should have value infinite.

### Value

A list with the following components:

times	the location of the hazard jumps.
wts	the jump size of hazard function at those locations.
lambda	the final value of the Lagrange multiplier.
"-2LLR"	The -2Log Likelihood ratio.
Pval	P-value
niters	number of iterations used

### Author(s)

Mai Zhou

### References

Zhou and Fang (2001). “Empirical likelihood ratio for 2 sample problems for censored data”. *Tech Report, Univ. of Kentucky, Dept of Statistics*

**Examples**

```

if(require("boot", quietly = TRUE)) {
###library(boot)
data(channing)
ymale <- channing[1:97,2]
dmale <- channing[1:97,5]
xmale <- channing[1:97,3]
yfemal <- channing[98:462,2]
dfemal <- channing[98:462,5]
xfemal <- channing[98:462,3]
fun1 <- function(x) { as.numeric(x <= 960) }
emplikH.disc2(x1=xfemal, d1=dfemal, y1=yfemal,
x2=xmale, d2=dmale, y2=ymale, theta=0.2, fun1=fun1, fun2=fun1, maxi=4, mini=-10)
#####
### You should get "-2LLR" = 1.511239 and a lot more other outputs.
#####
emplikH.disc2(x1=xfemal, d1=dfemal, y1=yfemal,
x2=xmale, d2=dmale, y2=ymale, theta=0.25, fun1=fun1, fun2=fun1, maxi=4, mini=-5)
#####
### This time you get "-2LLR" = 1.150098 etc. etc.
#####
}

```

emplikH1.test

*Empirical likelihood for hazard with right censored, left truncated data***Description**

Use empirical likelihood ratio and Wilks theorem to test the null hypothesis that

$$\int f(t)dH(t) = \theta$$

with right censored, left truncated data. Where  $H(t)$  is the unknown cumulative hazard function;  $f(t)$  can be any given function and  $\theta$  a given constant. In fact,  $f(t)$  can even be data dependent, just have to be 'predictable'.

**Usage**

```
emplikH1.test(x, d, y= -Inf, theta, fun, tola=.Machine$double.eps^.5)
```

**Arguments**

**x** a vector of the censored survival times.  
**d** a vector of the censoring indicators, 1-uncensor; 0-censor.  
**y** a vector of the observed left truncation times.  
**theta** a real number used in the  $H_0$  to set the hazard to this value.

fun	a left continuous (weight) function used to calculate the weighted hazard in $H_0$ . fun must be able to take a vector input. See example below.
tola	an optional positive real number specifying the tolerance of iteration error in solve the non-linear equation needed in constrained maximization.

### Details

This function is designed for the case where the true distributions are all continuous. So there should be no tie in the data.

The log empirical likelihood used here is the ‘Poisson’ version empirical likelihood:

$$\sum_{i=1}^n \delta_i \log(dH(x_i)) - [H(x_i) - H(y_i)] .$$

If there are ties in the data that are resulted from rounding, you may break the tie by adding a different tiny number to the tied observation(s). If those are true ties (thus the true distribution is discrete) we recommend use `emplikdisc.test()`.

The constant theta must be inside the so called feasible region for the computation to continue. This is similar to the requirement that in testing the value of the mean, the value must be inside the convex hull of the observations. It is always true that the NPMLE values are feasible. So when the computation complains that there is no hazard function satisfy the constraint, you should try to move the theta value closer to the NPMLE. When the computation stops prematurely, the -2LLR should have value infinite.

### Value

A list with the following components:

times	the location of the hazard jumps.
wts	the jump size of hazard function at those locations.
lambda	the Lagrange multiplier.
"-2LLR"	the -2Log Likelihood ratio.
Pval	P-value
niters	number of iterations used

### Author(s)

Mai Zhou

### References

Pan, X. and Zhou, M. (2002), “Empirical likelihood in terms of hazard for censored data”. *Journal of Multivariate Analysis* **80**, 166-188.

**Examples**

```
fun <- function(x) { as.numeric(x <= 6.5) }
emplikH1.test( x=c(1,2,3,4,5), d=c(1,1,0,1,1), theta=2, fun=fun)
fun2 <- function(x) {exp(-x)}
emplikH1.test( x=c(1,2,3,4,5), d=c(1,1,0,1,1), theta=0.2, fun=fun2)
```

---

emplikH2.test	<i>Empirical likelihood for weighted hazard with right censored, left truncated data</i>
---------------	------------------------------------------------------------------------------------------

---

**Description**

Use empirical likelihood ratio and Wilks theorem to test the null hypothesis that

$$\int f(t, \dots) dH(t) = K$$

with right censored, left truncated data, where  $H(t)$  is the (unknown) cumulative hazard function;  $f(t, \dots)$  can be any given left continuous function in  $t$ ; (of course the integral must be finite).

**Usage**

```
emplikH2.test(x, d, y= -Inf, K, fun, tola=.Machine$double.eps^.5, ...)
```

**Arguments**

x	a vector containing the censored survival times.
d	a vector of the censoring indicators, 1-uncensor; 0-censor.
y	a vector containing the left truncation times. If left as default value, -Inf, it means no truncation.
K	a real number used in the constraint, i.e. to set the weighted integral of hazard to this value.
fun	a left continuous (in $t$ ) weight function used to calculate the weighted hazard in $H_0$ . $\text{fun}(t, \dots)$ must be able to take a vector input $t$ .
tol	an optional positive real number specifying the tolerance of iteration error in solve the non-linear equation needed in constrained maximization.
...	additional parameter(s), if any, passing along to fun. This allows an implicit function of fun.

**Details**

This version works for implicit function  $f(t, \dots)$ .

This function is designed for continuous distributions. Thus we do not expect tie in the observation  $x$ . If you believe the true underlying distribution is continuous but the sample observations have tie due to rounding, then you might want to add a small number to the observations to break tie.

The likelihood used here is the ‘Poisson’ version of the empirical likelihood

$$\prod_{i=1}^n (dH(x_i))^{\delta_i} \exp[-H(x_i) + H(y_i)].$$

For discrete distributions we recommend use `emplikdisc.test()`.

Please note here the largest observed time is NOT automatically defined to be uncensored. In the `el.cen.EM()`, it is (to make F a proper distribution always).

The constant K must be inside the so called feasible region for the computation to continue. This is similar to the requirement that when testing the value of the mean, the value must be inside the convex hull of the observations for the computation to continue. It is always true that the NPMLE value is feasible. So when the computation cannot continue, that means there is no hazard function dominated by the Nelson-Aalen estimator satisfy the constraint. You may try to move the theta and K closer to the NPMLE. When the computation cannot continue, the -2LLR should have value infinite.

### Value

A list with the following components:

<code>times</code>	the location of the hazard jumps.
<code>wts</code>	the jump size of hazard function at those locations.
<code>lambda</code>	the Lagrange multiplier.
<code>"-2LLR"</code>	the -2Log Likelihood ratio.
<code>Pval</code>	P-value
<code>niters</code>	number of iterations used

### Author(s)

Mai Zhou

### References

Pan, XR and Zhou, M. (2002), “Empirical likelihood in terms of cumulative hazard for censored data”. *Journal of Multivariate Analysis* **80**, 166-188.

### See Also

`emplikHs.test2`

### Examples

```
z1<-c(1,2,3,4,5)
d1<-c(1,1,0,1,1)
fun4 <- function(x, theta) { as.numeric(x <= theta) }
emplikH2.test(x=z1,d=d1, K=0.5, fun=fun4, theta=3.5)
#Next, test if H(3.5) = log(2) .
emplikH2.test(x=z1,d=d1, K=log(2), fun=fun4, theta=3.5)
```



```
#Next, try one sample log rank test
indi <- function(x,y){ as.numeric(x >= y) }
fun3 <- function(t,z){rowsum(outer(z,t,FUN="indi"),group=rep(1,length(z)))}
emplikH2.test(x=z1, d=d1, K=sum(0.25*z1), fun=fun3, z=z1)
##this is testing if the data is from an exp(0.25) population.
```

emplikHs.disc2

*Two sample empirical likelihood ratio for discrete hazards with right censored, left truncated data. Many constraints.*

### Description

Use empirical likelihood ratio and Wilks theorem to test the null hypothesis that

$$\int f_1(t)I_{[dH_1 < 1]} \log(1 - dH_1(t)) - \int f_2(t)I_{[dH_2 < 1]} \log(1 - dH_2(t)) = \theta$$

where  $H_*(t)$  are the (unknown) discrete cumulative hazard functions;  $f_*(t)$  can be any predictable functions of  $t$ .  $\theta$  is a vector of parameters ( $\dim=q \geq 1$ ). The given value of  $\theta$  in these computation are the value to be tested. The data can be right censored and left truncated.

When the given constants  $\theta$  is too far away from the NPMLE, there will be no hazard function satisfy this constraint and the -2 Log empirical likelihood ratio will be infinite. In this case the computation will stop.

### Usage

```
emplikHs.disc2(x1, d1, y1= -Inf, x2, d2, y2 = -Inf,
  theta, fun1, fun2, maxit=25,tola = 1e-6, itertrace =FALSE)
```

### Arguments

x1	a vector, the observed survival times, sample 1.
d1	a vector, the censoring indicators, 1-uncensor; 0-censor.
y1	optional vector, the left truncation times.
x2	a vector, the observed survival times, sample 2.
d2	a vector, the censoring indicators, 1-uncensor; 0-censor.
y2	optional vector, the left truncation times.
fun1	a predictable function used to calculate the weighted discrete hazard in $H_0$ . fun1(x) must be able to take a vector input (length n) x, and output a matrix of n x q.
fun2	Ditto.
tola	an optional positive real number, the tolerance of iteration error in solve the non-linear equation needed in constrained maximization.
theta	a given vector of length q. for Ho constraint.
maxit	integer, maximum number of iteration.
itertrace	Logocal, lower bound for lambda

## Details

The log empirical likelihood been maximized is the ‘binomial empirical likelihood’:

$$\sum D_{1i} \log w_i + (R_{1i} - D_{1i}) \log[1 - w_i] + \sum D_{2j} \log v_j + (R_{2j} - D_{2j}) \log[1 - v_j]$$

where  $w_i = \Delta H_1(t_i)$  is the jump of the cumulative hazard function at  $t_i$ ,  $D_{1i}$  is the number of failures observed at  $t_i$ , and  $R_{1i}$  is the number of subjects at risk at time  $t_i$  (for sample one). Similar for sample two.

For discrete distributions, the jump size of the cumulative hazard at the last jump is always 1. We have to exclude this jump from the summation in the constraint calculation since  $\log(1 - dH(\cdot))$  do not make sense. In the likelihood, this term contribute a zero ( $0 * \text{Inf}$ ).

This function can handle multiple constraints. So  $\dim(\text{theta}) = q$ . The constants theta must be inside the so called feasible region for the computation to continue. This is similar to the requirement that in testing the value of the mean, the value must be inside the convex hull of the observations. It is always true that the NPMLE values are feasible. So when the computation stops, try move the theta closer to the NPMLE. When the computation stops, the -2LLR should have value infinite.

This code can also be used to compute one sample problems. You need to artificially supply data for sample two (with minimal sample size  $(2q+2)$ ), and supply a function fun2 that ALWAYS returns zero (zero vector or zero matrix). In the output, read the -2LLR(sample1).

## Value

A list with the following components:

times1	the location of the hazard jumps in sample 1.
times2	the location of the hazard jumps in sample 2.
lambda	the final value of the Lagrange multiplier.
"-2LLR"	The -2Log Likelihood ratio.
"-2LLR(sample1)"	The -2Log Likelihood ratio for sample 1 only.
niters	number of iterations used

## Author(s)

Mai Zhou

## References

Zhou and Fang (2001). “Empirical likelihood ratio for 2 sample problems for censored data”. *Tech Report, Univ. of Kentucky, Dept of Statistics*

## Examples

```
if(require("boot", quietly = TRUE)) {
  ###library(boot)
  data(channing)
  ymale <- channing[1:97,2]
```

```

dmale <- channing[1:97,5]
xmale <- channing[1:97,3]
yfemal <- channing[98:462,2]
dfemal <- channing[98:462,5]
xfemal <- channing[98:462,3]
fun1 <- function(x) { as.numeric(x <= 960) }
#####
emplikHs.disc2(x1=xfemal, d1=dfemal, y1=yfemal,
  x2=xmale, d2=dmale, y2=ymale, theta=0.25, fun1=fun1, fun2=fun1)
#####
### This time you get "-2LLR" = 1.150098 etc. etc.
#####
fun2 <- function(x){ cbind(as.numeric(x <= 960), as.numeric(x <= 860))}
##### fun2 has matrix output #####
emplikHs.disc2(x1=xfemal, d1=dfemal, y1=yfemal,
  x2=xmale, d2=dmale, y2=ymale, theta=c(0.25,0), fun1=fun2, fun2=fun2)
##### you get "-2LLR" = 1.554386, etc #####
}

```

---

emplikHs.test2	<i>Two sample empirical likelihood ratio test for hazards with right censored, left truncated data. Many constraints.</i>
----------------	---------------------------------------------------------------------------------------------------------------------------

---

**Description**

Use empirical likelihood ratio and Wilks theorem to test the null hypothesis that

$$\int f_1(t)dH_1(t) - \int f_2(t)dH_2(t) = \theta$$

where  $H_*(t)$  is the (unknown) cumulative hazard functions;  $f_*(t)$  can be any predictable functions of  $t$ .  $\theta$  is a vector of parameters (dim=q). The given value of  $\theta$  in these computation are the value to be tested. The data can be right censored and left truncated.

When the given constants  $\theta$  is too far away from the NPMLE, there will be no hazard function satisfy this constraint and the -2 Log empirical likelihood ratio will be infinite. In this case the computation will stop.

**Usage**

```

emplikHs.test2(x1, d1, y1= -Inf, x2, d2, y2 = -Inf,
  theta, fun1, fun2, maxit=25,tola = 1e-7,itertrace =FALSE)

```

**Arguments**

- x1                    a vector of length n1, the observed survival times, sample 1.
- d1                    a vector, the censoring indicators, 1-uncensor; 0-censor.
- y1                    optional vector, the left truncation times.
- x2                    a vector of length n2, the observed survival times, sample 2.

d2	a vector, the censoring indicators, 1-uncensor; 0-censor.
y2	optional vector, the left truncation times.
fun1	a predictable function used to calculate the weighted discrete hazard to form the null hypothesis $H_0$ . fun1(x) must be able to take a vector input (length n1) x, and output a matrix of n1 x q. When q=1, the output can also be a vector.
fun2	Ditto. but for length n2
tola	an optional positive real number, the tolerance of iteration error in solve the non-linear equation needed in constrained maximization.
theta	a given vector of length q. for Ho constraint.
maxit	integer, maximum number of Newton-Raphson type iterations.
itertrace	Logical, if the results of each iteration needs to be printed.

### Details

The log likelihood been maximized is the Poisson likelihood:

$$\sum D_{1i} \log w_i - \sum R_{1i} w_i + \sum D_{2j} \log v_j - \sum R_{2j} v_j$$

where  $w_i = \Delta H_1(t_i)$  is the jump of the cumulative hazard function at  $t_i$  (for first sample),  $D_{1i}$  is the number of failures observed at  $t_i$ ,  $R_{1i}$  is the number of subjects at risk at time  $t_i$ . Dido for sample two.

For (proper) discrete distributions, the jump size of the cumulative hazard at the last jump is always 1. So, in the likelihood ratio, it cancels. But the last jump of size 1 still matter when computing the constraint.

The constants theta must be inside the so called feasible region for the computation to continue. This is similar to the requirement that in testing the value of the mean, the value must be inside the convex hull of the observations. It is always true that the NPMLE values are feasible. So when the computation stops, try move the theta closer to the NPMLE, which we print out first thing in this function, even when other later computations do not go. When the computation stops, the -2LLR should have value infinite.

You can also use this function for one sample problems. You need to artificially supply data for sample two of minimal size (like size  $2q+2$ ), and specify a fun2() that ALWAYS return 0's (zero vector, with length=n2 vector length, or zero matrix, with dim n2 x q as the input). Then, look for -2LLR(sample1) in the output.

### Value

A list with the following components:

"-2LLR"	The -2Log empirical Likelihood ratio.
lambda	the final value of the Lagrange multiplier.
"-2LLR(sample1)"	The -2Log empirical likelihood ratio for sample one only. Useful in one sample problems.
"Llog(sample1)"	The numerator only of the above "-2LLR(sample1)", without -2.

**Author(s)**

Mai Zhou

**References**

Zhou and Fang (2001). "Empirical likelihood ratio for 2 sample problems for censored data". *Tech Report, Univ. of Kentucky, Dept of Statistics*

**See Also**

emplikH2.test

**Examples**

```

if(require("boot", quietly = TRUE)) {
###library(boot)
data(channing)
ymale <- channing[1:97,2]
dmale <- channing[1:97,5]
xmale <- channing[1:97,3]
yfemal <- channing[98:462,2]
dfemal <- channing[98:462,5]
xfemal <- channing[98:462,3]
fun1 <- function(x) { as.numeric(x <= 960) }
#####
fun2 <- function(x){ cbind(as.numeric(x <= 960), as.numeric(x <= 860))}
##### fun2 has matrix output #####
emplikHs.test2(x1=xfemal, d1=dfemal, y1=yfemal,
  x2=xmale, d2=dmale, y2=ymale, theta=c(0,0), fun1=fun1, fun2=fun2)
}
#####
##### Second example:
if(require("KMsurv", quietly = TRUE)) {
###library(KMsurv)
data(kidney)
### these functions counts the risk set size, so delta=1 always ###
temp1 <- Wdataclean3(z=kidney$time[kidney[,3]==1], d=rep(1,43) )
temp2 <- DnR(x=temp1$value, d=temp1$dd, w=temp1$weight)
TIME <- temp2$times
RISK <- temp2$n.risk
fR1 <- approxfun(x=TIME, y=RISK, method="constant", yright=0, rule=2, f=1)
temp1 <- Wdataclean3(z=kidney$time[kidney[,3]==2], d=rep(1,76) )
temp2 <- DnR(x=temp1$value, d=temp1$dd, w=temp1$weight)
TIME <- temp2$times
RISK <- temp2$n.risk
fR2 <- approxfun(x=TIME, y=RISK, method="constant", yright=0, rule=2, f=1)

### the weight function for two sample Gehan-Wilcoxon type test ###
fun <- function(t){ fR1(t)*fR2(t)/((76*43)*sqrt(119/(76*43)) )}
### Here comes the test: ###
emplikHs.test2(x1=kidney[kidney[,3]==1,1],d1=kidney[kidney[,3]==1,2],
  x2=kidney[kidney[,3]==2,1],d2=kidney[kidney[,3]==2,2],

```

```

    theta=0, fun1= fun, fun2=fun)
### The results should include this ###
#"$-2LLR"
#[1] 0.002473070
#
#$lambda
#[1] -0.1713749
#####
##### the weight function for log-rank test #####
funlogrank <- function(t){sqrt(119/(76*43))*fR1(t)*fR2(t)/(fR1(t)+fR2(t))}
##### Now the log-rank test ###
emplikHs.test2(x1=kidney[kidney[,3]==1,1],d1=kidney[kidney[,3]==1,2],
  x2=kidney[kidney[,3]==2,1],d2=kidney[kidney[,3]==2,2],
  theta=0, fun1=funlogrank, fun2=funlogrank)
##### The result of log rank test should include this ###
#
#"$-2LLR"
#[1] 2.655808
#
#$lambda
#[1] 3.568833
#####
##### the weight function for both type test #####
funBOTH <- function(t) {
  cbind(sqrt(119/(76*43))*fR1(t)*fR2(t)/(fR1(t)+fR2(t)),
    fR1(t)*fR2(t)/((76*43)*sqrt(119/(76*43)))) }
##### The test that combine both tests ###
emplikHs.test2(x1=kidney[kidney$type==1,1],d1=kidney[kidney$type==1,2],
  x2=kidney[kidney$type==2,1],d2=kidney[kidney$type==2,2],
  theta=c(0,0), fun1=funBOTH, fun2=funBOTH)
##### the result should include this ###
#
#"$-2LLR"
#[1] 13.25476
#
#$lambda
#[1] 14.80228 -21.86733
#####
}

```

---

findUL

---

*Find the Wilks Confidence Interval from the Given (empirical) Likelihood Ratio Function*


---

## Description

This program uses `uniroot()` to find the upper and lower (Wilks) confidence limits based on the -2 log likelihood ratio, which the required input `fun` is supposed to supply.

Basically, starting from MLE, we search on both directions, by step away from MLE, until we find values that have  $-2LLR = \text{level}$ . (the value of  $-2LLR$  at MLE is supposed to be zero.)

At current implementation, only handles one dimensional parameter, i.e. only confidence intervals, not confidence regions.

For examples of using this function to find confidence interval, see the pdf vignettes file.

### Usage

```
findUL (step = 0.01, initStep = 0, fun, MLE, level = 3.84, ...)
```

### Arguments

step	a positive number. The starting step size of the search. Reasonable value should be about 1/5 of the SD of MLE.
initStep	a nonnegative number. The first step size of the search. Sometimes, you may want to put a larger innitStep to speed the search.
fun	a function that returns a list. One of the item in the list should be "-2LLR", which is the -2 log (empirical) likelihood ratio. The first input of fun must be the parameter for which we are seeking the confidence interval. (The MLE or NPMLE of this parameter should be supplied as in the input MLE). The rest of the input to fun are typically the data. If the first input of fun is set to MLE, then the returned -2LLR should be 0.
MLE	The MLE of the parameter. No need to be exact, as long as it is inside the confidence interval.
level	an optional positive number, controls the confidence level. Default to 3.84 = $\text{chisq}(0.95, \text{df}=1)$ . Change to 2.70= $\text{chisq}(0.90, \text{df}=1)$ to get a 90% confidence interval.
...	additional arguments, if any, to pass to fun.

### Details

Basically we repeatedly testing the value of the parameter, until we find those which the -2 log likelihood value is equal to 3.84 (or other level, if set differently).

If there is no value exactly equal to 3.84, we stop at the value which result a -2 log likelihood just below 3.84. (as in the discrete case, like quantiles.)

### Value

A list with the following components:

Low	the lower limit of the confidence interval.
Up	the upper limit of the confidence interval.
FstepL	the final step size when search lower limit. An indication of the precision.
FstepU	Ditto. An indication of the precision of the upper limit.
Lvalue	The -2LLR value of the final Low value. Should be approximately equal to level. If larger than level, than the confidence interval limit Low is wrong.
Uvalue	Ditto. Should be approximately equa to level.

**Author(s)**

Mai Zhou

**References**

Zhou, M. (2016). Empirical Likelihood Method in Survival Analysis. CRC Press.

**Examples**

```
## example with tied observations. Kaplan-Meier mean=4.0659.
## For more examples see vignettes.
x <- c(1, 1.5, 2, 3, 4, 5, 6, 5, 4, 1, 2, 4.5)
d <- c(1, 1, 0, 1, 0, 1, 1, 1, 1, 0, 0, 1)
myfun6 <- function(theta, x, d) {
  el.cen.EM2(x, d, fun=function(t){t}, mu=theta)
}
findUL(step=0.2, fun=myfun6, MLE=4.0659, x=x, d=d)
```

---

 findUL2

---

*Find the Wilks Confidence Interval from the Given (empirical) Likelihood Ratio Function*


---

**Description**

This program uses simple search and uniroot( ) to find the upper and lower (Wilks) confidence limits based on the -2 log likelihood ratio, which the required input fun is supposed to supply.

This function is faster than findUL( ).

Basically, starting from MLE, we search on both directions, by step away from MLE, until we find values that have -2LLR = level. (the value of -2LLR at MLE is supposed to be zero.)

At current implementation, only handles one dimensional parameter, i.e. only confidence intervals, not confidence regions.

For examples of using this function to find confidence interval, see the pdf vignettes file.

**Usage**

```
findUL2(step=0.01, initStep=0, fun, MLE, level=3.84, tol=.Machine$double.eps^0.5, ...)
```

**Arguments**

step	a positive number. The starting step size of the search. Reasonable value should be about 1/5 of the SD of MLE.
initStep	a nonnegative number. The first step size of the search. Sometimes, you may want to put a larger innitStep to speed the search.



fun	a function that returns a list. One of the item in the list should be "-2LLR", which is the -2 log (empirical) likelihood ratio. The first input of fun must be the parameter for which we are seeking the confidence interval. (The MLE or NPMLE of this parameter should be supplied as in the input MLE). The rest of the input to fun are typically the data. If the first input of fun is set to MLE, then the returned -2LLR should be 0.
MLE	The MLE of the parameter. No need to be exact, as long as it is inside the confidence interval.
level	an optional positive number, controls the confidence level. Default to 3.84 = $\text{chisq}(0.95, \text{df}=1)$ . Change to 2.70= $\text{chisq}(0.90, \text{df}=1)$ to get a 90% confidence interval.
tol	tolerance to pass to uniroot( ). Default to $.\text{Machine}\$\text{double.eps}^0.5$
...	additional arguments, if any, to pass to fun.

### Details

Basically we repeatedly testing the value of the parameter, until we find those which the -2 log likelihood value is equal to 3.84 (or other level, if set differently).

If there is no value exactly equal to 3.84, we stop at the value which result a -2 log likelihood just below 3.84. (as in the discrete case, like quantiles.)

### Value

A list with the following components:

Low	the lower limit of the confidence interval.
Up	the upper limit of the confidence interval.
FstepL	the final step size when search lower limit. An indication of the precision.
FstepU	Ditto. An indication of the precision of the upper limit.
Lvalue	The -2LLR value of the final Low value. Should be approximately equal to level. If larger than level, than the confidence interval limit Low is wrong.
Uvalue	Ditto. Should be approximately equa to level.

### Author(s)

Mai Zhou

### References

- Zhou, M. (2016). Empirical Likelihood Method in Survival Analysis. CRC Press.
- Zhou, M. (2002). Computing censored empirical likelihood ratio by EM algorithm. *JCGS*

**Examples**

```
## example with tied observations. Kaplan-Meier mean=4.0659.
## For more examples see vignettes.
x <- c(1, 1.5, 2, 3, 4, 5, 6, 5, 4, 1, 2, 4.5)
d <- c(1, 1, 0, 1, 0, 1, 1, 1, 1, 0, 0, 1)
myfun6 <- function(theta, x, d) {
  el.cen.EM2(x, d, fun=function(t){t}, mu=theta)
}
findUL2(step=0.2, fun=myfun6, MLE=4.0659, x=x, d=d)
```

findULold

*Find the Wilks Confidence Interval from the Given (empirical) Likelihood Ratio Function*

**Description**

This program uses simple search to find the upper and lower (Wilks) confidence limits based on the  $-2 \log$  likelihood ratio, which the required input fun is supposed to supply.

Basically, starting from MLE, we search on both directions, by step away from MLE, until we find values that have  $-2LLR = \text{level}$ . (the value of  $-2LLR$  at MLE is supposed to be zero.)

At current implementation, only handles one dimensional parameter, i.e. only confidence intervals, not confidence regions.

For examples of using this function to find confidence interval, see the pdf vignettes file.

**Usage**

```
findULold (step = 0.01, initStep = 0, fun, MLE, level = 3.84, ...)
```

**Arguments**

step	a positive number. The starting step size of the search. Reasonable value should be about 1/5 of the SD of MLE.
initStep	a nonnegative number. The first step size of the search. Sometimes, you may want to put a larger innitStep to speed the search.
fun	a function that returns a list. One of the item in the list should be "-2LLR", which is the $-2 \log$ (empirical) likelihood ratio. The first input of fun must be the parameter for which we are seeking the confidence interval. (The MLE or NPMLE of this parameter should be supplied as in the input MLE). The rest of the input to fun are typically the data. If the first input of fun is set to MLE, then the returned $-2LLR$ should be 0.
MLE	The MLE of the parameter. No need to be exact, as long as it is inside the confidence interval.
level	an optional positive number, controls the confidence level. Default to 3.84 = $\text{chisq}(0.95, \text{df}=1)$ . Change to 2.70= $\text{chisq}(0.90, \text{df}=1)$ to get a 90% confidence interval.
...	additional arguments, if any, to pass to fun.

**Details**

Basically we repeatedly testing the value of the parameter, until we find those which the -2 log likelihood value is equal to 3.84 (or other level, if set differently).

If there is no value exactly equal to 3.84, we stop at the value which result a -2 log likelihood just below 3.84. (as in the discrete case, like quantiles.)

**Value**

A list with the following components:

Low	the lower limit of the confidence interval.
Up	the upper limit of the confidence interval.
FstepL	the final step size when search lower limit. An indication of the precision.
FstepU	Ditto. An indication of the precision of the upper limit.
Lvalue	The -2LLR value of the final Low value. Should be approximately equal to level. If larger than level, than the confidence interval limit Low is wrong.
Uvalue	Ditto. Should be approximately equa to level.

**Author(s)**

Mai Zhou

**References**

Zhou, M. (2016). Empirical Likelihood Method in Survival Analysis. CRC Press.

**Examples**

```
## example with tied observations. Kaplan-Meier mean=4.0659.
## For more examples see vignettes.
x <- c(1, 1.5, 2, 3, 4, 5, 6, 5, 4, 1, 2, 4.5)
d <- c(1, 1, 0, 1, 0, 1, 1, 1, 1, 0, 0, 1)
myfun6 <- function(theta, x, d) {
  el.cen.EM2(x, d, fun=function(t){t}, mu=theta)
}
findULold(step=0.2, fun=myfun6, MLE=4.0659, level = qchisq(0.9, df=1) , x=x, d=d)
```

### Description

Krall, Uthoff, and Harley (1975) analyzed data from a study on multiple myeloma in which researchers treated 65 patients with alkylating agents.

Of those patients, 48 died during the study and 17 survived. In the data set MYELOMA, the variable TIME represents the survival time in months from diagnosis. The variable VSTATUS consists of two values, 0 and 1, indicating whether the patient was alive or dead, respectively, at the of end the study. If the value of VSTATUS is 0, the corresponding value of TIME is censored.

The variables thought to be related to survival are LOGBUN (log BUN at diagnosis), HGB (hemoglobin at diagnosis), PLATELET (platelets at diagnosis: 0=abnormal, 1=normal), AGE (age at diagnosis in years), LOGWBC (log WBC at diagnosis), FRAC (fractures at diagnosis: 0=none, 1=present), LOGPBM (log percentage of plasma cells in bone marrow), PROTEIN (proteinuria at diagnosis), and SCALC (serum calcium at diagnosis).

Data are from [http://ftp.sas.com/techsup/download/sample/samp\\_lib/statsampExamples\\_of\\_Coxs\\_Model.html](http://ftp.sas.com/techsup/download/sample/samp_lib/statsampExamples_of_Coxs_Model.html)

### Usage

```
data(myeloma)
```

### Format

A data frame containing 65 observations on 11 variables:

```
[,1] "time"
[,2] "vstatus"
[,3] "logBUN"
[,4] "HGB"
[,5] "platelet"
[,6] "age"
[,7] "logWBC"
[,8] "FRAC"
[,9] "logPBM"
[,10] "protein"
[,11] "SCALC"
```

### References

Krall, J.M., Uthoff, V.A., and Harley, J. B. (1975), A Step-up Procedure for Selecting Variables Associated with Survival, *Biometrics*, 31, 49-57.

**Description**

Use the empirical likelihood ratio and Wilks theorem to test if the regression coefficient is equal to beta, based on the rank estimator for the AFT model.

The log empirical likelihood been maximized is

$$\sum_{d=1} \log \Delta F(e_i) + \sum_{d=0} \log[1 - F(e_i)];$$

where  $e_i$  are the residuals.

**Usage**

```
RankRegTest(y, d, x, beta, type="Gehan")
```

**Arguments**

y	a vector of length N, containing the censored responses.
d	a vector (length N) of either 1's or 0's. d=1 means y is uncensored; d=0 means y is right censored.
x	a matrix of size N by q.
beta	a vector of length q. the value of the regression coefficient to be tested in the model $y_i = \beta x_i + \epsilon_i$ .
type	default to Gehan type. The other option is Logrank type.

**Details**

The estimator of beta can be obtained by function `rankaft( )` in the package `rankreg`. But here you may test other values of beta. If you test the beta value that is obtained from the `rankaft( )`, then the -2LLR should be 0 and the p-value should be 1.

The above likelihood should be understood as the likelihood of the error term, so in the regression model the error  $e_i$  should be iid.

The estimation equation used when maximize the empirical likelihood is

$$0 = \sum_i \phi(e_i) d_i \Delta F(e_i) (x_i - \bar{x}_i) / (n w_i)$$

which was described in detail in the references below.

See also the function `RankRegTestH`, which is based on the hazard likelihood.

**Value**

A list with the following components:

"-2LLR"	the -2 loglikelihood ratio; should have approximate chisq distribution under $H_0$ .
logel2	the log empirical likelihood, under estimating equation.
logel	the log empirical likelihood of the Kaplan-Meier of e's.
prob	the probabilities that max the empirical likelihood under rank estimating equation constraint.

**Author(s)**

Mai Zhou.

**References**

Kalbfleisch, J. and Prentice, R. (2002) The Statistical Analysis of Failure Time Data. 2nd Ed. Wiley, New York. (Chapter 7)

Jin, Z., Lin, D.Y., Wei, L. J. and Ying, Z. (2003). Rank-based inference for the accelerated failure time model. *Biometrika*, **90**, 341-53.

Zhou, M. (2005). Empirical likelihood analysis of the rank estimator for the censored accelerated failure time model. *Biometrika*, **92**, 492-98.

**Examples**

```
data(myeloma)
RankRegTest(y=myeloma[,1], d=myeloma[,2], x=myeloma[,3], beta= -15.50147)
# you should get "-2LLR" = 9.050426e-05 (practically zero)
# The beta value, -15.50147, was obtained by rankaft() from the rankreg package.
```

---

RankRegTestH

*Test the AFT model, Rank Regression estimator by (Hazard)Empirical Likelihood*

---

**Description**

Use the empirical likelihood ratio and Wilks theorem to test if the regression coefficient is equal to beta, based on the rank estimator/estimating equation of the AFT model.

The log empirical likelihood been maximized is the hazard empirical likelihood.

**Usage**

```
RankRegTestH(y, d, x, beta, type="Gehan")
```

**Arguments**

y	a vector of length N, containing the censored responses.
d	a vector (length N) of either 1's or 0's. d=1 means y is uncensored; d=0 means y is right censored.
x	a matrix of size N by q.
beta	a vector of length q. the value of the regression coefficient to be tested in the model $y_i = \beta x_i + \epsilon_i$ .
type	default to Gehan type. The other option is Logrank type.

## Details

The estimator of beta can be obtained by function `rankaft( )` in the package `rankreg`. But here you may test other values of beta. If you test the beta value that is obtained from the `rankaft( )`, then the -2LLR should be 0 and the p-value should be 1.

The above likelihood should be understood as the likelihood of the error term, so in the regression model the error  $e_i$  should be iid.

The estimating equation used when maximize the empirical likelihood is

$$0 = \sum_i R(e_i)\phi(e_i)d_i\Delta A(e_i)(x_i - \bar{x}_i)$$

where all notation was described in detail in the references below.

## Value

A list with the following components:

"-2LLR"	the -2 loglikelihood ratio; should have approximate chisq distribution under $H_0$ .
logel2	the log empirical likelihood, under estimating equation.
logel	the log empirical likelihood of the Kaplan-Meier of e's.

## Author(s)

Mai Zhou

## References

Zhou, M. (2016) Empirical Likelihood Methods in Survival Analysis. CRC Press.

Kalbfleisch, J. and Prentice, R. (2002) The Statistical Analysis of Failure Time Data. 2nd Ed. Wiley, New York. (Chapter 7)

Jin, Z., Lin, D.Y., Wei, L. J. and Ying, Z. (2003). Rank-based inference for the accelerated failure time model. *Biometrika*, **90**, 341-53.

Zhou, M. (2005). Empirical likelihood analysis of the rank estimator for the censored accelerated failure time model. *Biometrika*, **92**, 492-498.

## Examples

```
data(myeloma)
RankRegTestH(y=myeloma[,1], d=myeloma[,2], x=myeloma[,3], beta= -15.50147)
# you should get "-2LLR" = 9.050426e-05 (practically zero)
# The beta value, -15.50147, was obtained by rankaft() from
# the rankreg package.
```

**Description**

Use empirical likelihood ratio to test the hypothesis  $H_0$ :  $(1-b_0)$ th quantile of sample 1 =  $(1-t_0)$ th quantile of sample 2. This is the same as testing  $H_0$ :  $R(t_0) = b_0$ , where  $R(\cdot)$  is the ROC curve.

The log empirical likelihood been maximized is

$$\sum_{d1=1} \log \Delta F_1(t1_i) + \sum_{d1=0} \log[1 - F_1(t1_i)] + \sum_{d2=1} \log \Delta F_2(t2_j) + \sum_{d2=0} \log[1 - F_2(t2_j)].$$

This empirical likelihood ratio has a chi square limit under  $H_0$ .

**Usage**

ROCnp(t1, d1, t2, d2, b0, t0)

**Arguments**

t1	a vector of length n. Observed times, may be right censored.
d1	a vector of length n, censoring status. d=1 means t is uncensored; d=0 means t is right censored.
t2	a vector of length m. Observed times, may be right censored.
d2	a vector of length m, censoring status.
b0	a scalar between 0 and 1.
t0	a scalar, between 0 and 1.

**Details**

Basically, we first test  $(1-b_0)$ th quantile of sample 1 = c and also test  $(1-t_0)$ th quantile of sample 2 = c. This way we obtain two log likelihood ratios.

Then we minimize the sum of the two log likelihood ratio over c.

See the tech report below for details on a similar setting.

**Value**

A list with the following components:

"-2LLR"	the -2 loglikelihood ratio; have approximate chisq distribution under $H_0$ .
cstar	the estimated common quantile.

**Author(s)**

Mai Zhou.



## References

- Zhou, M. and Liang, H (2008). Empirical Likelihood for Hybrid Two Sample Problem with Censored Data. *Univ. Kentucky Tech. Report*.
- Su, H., Zhou, M. and Liang, H. (2011). Semi-parametric hybrid empirical likelihood inference for two-sample comparison with censored data. *Lifetime Data Analysis*, **17**, 533-551.

## Examples

```
#### An example of testing the equality of two medians. No censoring.
ROCnp(t1=rexp(100), d1=rep(1,100), t2=rexp(120), d2=rep(1,120), b0=0.5, t0=0.5)
#####
#### Next, an example of finding 90 percent confidence interval of R(0.5)
#### Note: We are finding confidence interval for R(0.5). So we are testing
#### R(0.5)= 0.35, 0.36, 0.37, 0.38, etc. try to find values so that
#### testing R(0.5) = L , U has p-value of 0.10, then [L, U] is the 90 percent
#### confidence interval for R(0.5).
#set.seed(123)
#t1 <- rexp(200)
#t2 <- rexp(200)
#ROCnp( t1=t1, d1=rep(1, 200), t2=t2, d2=rep(1, 200), b0=0.5, t0=0.5)$"-2LLR"
#### since the -2LLR value is less than 2.705543 = qchisq(0.9, df=1), so the
#### confidence interval contains 0.5.
#gridpoints <- 35:65/100
#ELvalues <- gridpoints
#for( i in 1:31 ) ELvalues[i] <- ROCnp(t1=t1, d1=rep(1, 200),
#                                     t2=t2, d2=rep(1, 200), b0=gridpoints[i], t0=0.5)$"-2LLR"
#myfun1 <- approxfun(x=gridpoints, y=ELvalues)
#uniroot( f= function(x){myfun1(x)-2.705543}, interval= c(0.35, 0.5) )
#uniroot( f= function(x){myfun1(x)-2.705543}, interval= c(0.5, 0.65) )
#### So, taking the two roots, we see the 90 percent confidence interval for R(0.5)
#### in this case is [0.4478081, 0.5889425].
```

---

ROCnp2

*Test the ROC curve by Empirical Likelihood*

---

## Description

Use empirical likelihood ratio to test the hypothesis  $H_0$ :  $(1-b_0)$ th quantile of sample 1 =  $(1-t_0)$ th quantile of sample 2. This is the same as testing  $H_0$ :  $R(t_0)= b_0$ , where  $R(\cdot)$  is the ROC curve.

The log empirical likelihood been maximized is

$$\sum_{d1=1} \log \Delta F_1(t1_i) + \sum_{d1=0} \log[1 - F_1(t1_i)] + \sum_{d2=1} \log \Delta F_2(t2_j) + \sum_{d2=0} \log[1 - F_2(t2_j)].$$

This empirical likelihood ratio has a chi square limit under  $H_0$ .

## Usage

ROCnp2(t1, d1, t2, d2, b0, t0)

**Arguments**

t1	a vector of length n. Observed times, sample 1. may be right censored.
d1	a vector of length n, censoring status. d=1 means t is uncensored; d=0 means t is right censored.
t2	a vector of length m. Observed times, sample 2. may be right censored.
d2	a vector of length m, censoring status.
b0	a scalar, between 0 and 1.
t0	a scalar, between 0 and 1.

**Details**

First, we test  $(1-b_0)$ th quantile of sample 1 = c and also test  $(1-t_0)$ th quantile of sample 2 = c. This way we obtain two log likelihood ratios.

Then we minimize the sum of the two log likelihood ratios over c.

This version use an exhaust search for the minimum (over c). Since the objective (log lik) are piecewise constants, the optimum( ) function in R do not work well. See the tech report below for details on a similar setting.

**Value**

A list with the following components:

"-2LLR"	the -2 loglikelihood ratio; have approximate chisq distribution under $H_0$ .
cstar	the estimated common quantile.

**Author(s)**

Mai Zhou

**References**

Su, Haiyan; Zhou, Mai and Liang, Hua (2011). Semi-parametric Hybrid Empirical Likelihood Inference for Two Sample Comparison with Censored Data. *Lifetime Data Analysis*, **17**, 533-551.

**Examples**

```
#### An example of testing the equality of two medians.
#### No censoring.
# ROCnp2(t1=rexp(100), d1=rep(1,100), t2=rexp(120),
#                               d2=rep(1,120), b0=0.5, t0=0.5)
#####
#### This example do not work on the Solaris Sparc machine.
#### But works fine on other platforms.
#####
#### Next, an example of finding 90 percent confidence
#### interval of R(0.5)
#### Note: We are finding confidence interval for R(0.5).
#### So we are testing
```

```
##### R(0.5)= 0.35, 0.36, 0.37, 0.38, etc. try to find
##### values so that testing R(0.5) = L , U has p-value
##### of 0.10, then [L, U] is the 90 percent
##### confidence interval for R(0.5).
#set.seed(123)
#t1 <- rexp(200)
#t2 <- rexp(200)
#ROCnp( t1=t1, d1=rep(1, 200), t2=t2, d2=rep(1, 200),
#      b0=0.5, t0=0.5)$"-2LLR"
##### since the -2LLR value is less than
##### 2.705543 = qchisq(0.9, df=1), so the
##### confidence interval contains 0.5.
#gridpoints <- 35:65/100
#ELvalues <- gridpoints
#for(i in 1:31) ELvalues[i] <- ROCnp2(t1=t1, d1=rep(1, 200),
#  t2=t2, d2=rep(1, 200), b0=gridpoints[i], t0=0.5)$"-2LLR"
#myfun1 <- approxfun(x=gridpoints, y=ELvalues)
#uniroot(f=function(x){myfun1(x)-2.705543},
#      interval= c(0.35, 0.5) )
#uniroot(f= function(x){myfun1(x)-2.705543},
#      interval= c(0.5, 0.65) )
##### So, taking the two roots, we see the 90 percent
##### confidence interval for R(0.5) in this
##### case is [0.4457862, 0.5907723].
#####
```

---

smallcell

*Smallcell Lung Cancer Data*


---

## Description

There are 121 observations on 4 variables. Arm is the indication of two treatments. Entry is the age of the patient at entry. Survival is the survival time and indicator is the censoring indicator (right censoring). For more details please see the reference below.

Data are from Ying, Z., Jung, SH, and Wei, LJ (1995). Median regression analysis with censored data. *Journal of the American Statistical Association*, 90, 178-184.

## Usage

```
data(smallcell)
```

## Format

A data frame containing 121 observations on 4 variables:

```
[,1] "arm"
[,2] "entry"
[,3] "survival"
[,4] "indicator"
```

## References

Ying, Z., Jung, SH, and Wei, LJ (1995). Median regression analysis with censored data. Journal of the American Statistical Association, 90, 178-184.

---

WRegEst

*Compute the casewise weighted regression estimator for AFT model*

---

## Description

For the AFT model, this function computes the case weighted estimator of beta. Either the least squares estimator or the regression quantile estimator.

## Usage

```
WRegEst(x, y, delta, LS=TRUE, tau=0.5)
```

## Arguments

x	a matrix of size N by q.
y	a vector of length N, containing the censored responses. Usually the log of the original observed failure times.
delta	a vector (length N) of either 1's or 0's. d=1 means y is uncensored; d=0 means y is right censored.
LS	a logical value. If TRUE then the function will return the least squares estimator. If FALSE then the function will return the quantile regression estimator, with the quantile level specified by tau. .
tau	a scalar, between 0 and 1. The quantile to be used in quantile regression. If tau=0.5 then it is the median regression. If LS=TRUE, then it is ignored.

## Details

Due to the readily available minimizer, we only provide least squares and quantile regression here. However, in the companion testing function WRegTest the user can supply a self defined psi function, corresponding to the general M-estimation in the regression modeling. (since there is no minimization needed).

The estimator is the minimizer of

$$\sum_{i=1}^n w_i \rho(Y_i - X_i b)$$

Assuming a correlation model

$$Y_i = X_i \beta + \sigma(X_i) \epsilon_i$$

, where  $\rho()$  is either the square or the absolute value function.

## Value

The estimator  $\hat{\beta}$ .

**Author(s)**

Mai Zhou.

**References**

Zhou, M.; Bathke, A. and Kim, M. (2012). Empirical likelihood analysis of the Heteroscastic Accelerated Failure Time model. *Statistica Sinica*, **22**, 295-316.

**Examples**

```
data(smallcell)
WRegEst(x=cbind(1,smallcell[,1],smallcell[,2]),
        y=smallcell[,3], delta=smallcell[,4])
#####
#### you should get          x1          x2          x3
####                -59.22126 -488.41306   16.03259
#####
WRegEst(x=cbind(1,smallcell[,1],smallcell[,2]),
        y=log10(smallcell[,3]), delta=smallcell[,4], LS=FALSE)
#####
#### you should get
####      [1]      2.603342985  -0.263000044  0.003836832
#####
```

WRegTest

*Test the case weighted regression estimator by Empirical Likelihood***Description**

Use the empirical likelihood ratio and Wilks theorem to test if the regression coefficient is equal to  $\beta_0$ , by the case weighted estimation method.

The log empirical likelihood been maximized is

$$\sum_{d=1} \log \Delta F(y_i) + \sum_{d=0} \log [1 - F(y_i)].$$

**Usage**

```
WRegTest(x, y, delta, beta0, psifun=function(t){t})
```

**Arguments**

**x** a matrix of size N by q. Random design matrix.  
**y** a vector of length N, containing the censored responses.  
**delta** a vector (length N) of either 1's or 0's. delta=1 means y is uncensored; delta=0 means y is right censored.  
**beta0** a vector of length q. The value of the regression coefficient to be tested in the linear model .

`psifun` the estimating function. The definition of it determines the type of estimator under testing.

### Details

The above likelihood should be understood as the likelihood of the censored responses `y` and `delta`.

This version can handle the model where `beta` is a vector (of length `q`).

The estimation equations used when maximize the empirical likelihood is

$$0 = \sum \delta_i \Delta F(Y_i) X_i (Y_i - X_i \beta_0)$$

which was described in detail in the reference below.

For median regression (Least Absolute Deviation) estimator, you should define the `psifun` as `+1, -1` or `0` when `t` is `> 0, < 0` or `= 0`.

For ordinary least squares estimator, `psifun` should be the identity function `psifun <- function(t)t`.

### Value

A list with the following components:

"-2LLR" the -2 log likelihood ratio; have approximate chisq distribution under  $H_0$ .

P-val the p-value using the chi-square approximation.

### Author(s)

Mai Zhou.

### References

Zhou, M.; Bathke, A. and Kim, M. (2012). Empirical likelihood analysis of the case weighted estimator in heteroscastic AFT model. *Statistica Sinica*, **22**, 295-316.

### Examples

```
xx <- c(28, -44, 29, 30, 26, 27, 22, 23, 33, 16, 24, 29, 24, 40, 21, 31, 34, -2, 25, 19)
```

# Index

## \*Topic **datasets**

myeloma, [43](#)  
smallcell, [51](#)

## \*Topic **htest**

BJnoint, [2](#)  
bjtest, [3](#)  
bjtest1d, [5](#)  
bjtestII, [6](#)  
el.cen.EM, [8](#)  
el.cen.EM2, [10](#)  
el.cen.test, [14](#)  
el.ltrc.EM, [16](#)  
el.test, [18](#)  
el.test.wt, [20](#)  
el.test.wt2, [21](#)  
el.trun.test, [23](#)  
emplikH.disc, [25](#)  
emplikH.disc2, [27](#)  
emplikH1.test, [29](#)  
emplikH2.test, [31](#)  
emplikHs.disc2, [33](#)  
emplikHs.test2, [35](#)  
findUL, [38](#)  
findUL2, [40](#)  
findULold, [42](#)  
RankRegTest, [44](#)  
RankRegTestH, [46](#)  
ROCnp, [48](#)  
ROCnp2, [49](#)  
WRegTest, [53](#)

## \*Topic **nonparametric**

BJnoint, [2](#)  
bjtest, [3](#)  
bjtest1d, [5](#)  
bjtestII, [6](#)  
el.cen.EM, [8](#)  
el.cen.EM2, [10](#)  
el.cen.test, [14](#)  
el.ltrc.EM, [16](#)

el.test, [18](#)  
el.test.wt, [20](#)  
el.test.wt2, [21](#)  
el.trun.test, [23](#)  
emplikH.disc, [25](#)  
emplikH.disc2, [27](#)  
emplikH1.test, [29](#)  
emplikH2.test, [31](#)  
emplikHs.disc2, [33](#)  
emplikHs.test2, [35](#)  
findUL, [38](#)  
findUL2, [40](#)  
findULold, [42](#)  
RankRegTest, [44](#)  
RankRegTestH, [46](#)  
ROCnp, [48](#)  
ROCnp2, [49](#)  
WRegEst, [52](#)  
WRegTest, [53](#)

## \*Topic **survival**

el.cen.EM, [8](#)  
el.cen.EM2, [10](#)  
el.cen.test, [14](#)  
el.ltrc.EM, [16](#)  
el.trun.test, [23](#)  
emplikH.disc, [25](#)  
emplikH.disc2, [27](#)  
emplikH1.test, [29](#)  
emplikH2.test, [31](#)  
emplikHs.disc2, [33](#)  
emplikHs.test2, [35](#)

BJnoint, [2](#)  
bjtest, [3](#)  
bjtest1d, [5](#)  
bjtestII, [6](#)

el.cen.EM, [8](#)  
el.cen.EM2, [10](#)  
el.cen.test, [14](#)

el.ltrc.EM, [16](#)  
el.test, [18](#)  
el.test.wt, [20](#)  
el.test.wt2, [21](#)  
el.trun.test, [23](#)  
emplikH.disc, [25](#)  
emplikH.disc2, [27](#)  
emplikH1.test, [29](#)  
emplikH2.test, [31](#)  
emplikHs.disc2, [33](#)  
emplikHs.test2, [35](#)

findUL, [38](#)  
findUL2, [40](#)  
findULold, [42](#)

myeloma, [43](#)

RankRegTest, [44](#)  
RankRegTestH, [46](#)  
ROCnp, [48](#)  
ROCnp2, [49](#)

smallcell, [51](#)

WRegEst, [52](#)  
WRegTest, [53](#)