

# Package ‘kmc’

May 8, 2026

**Type** Package

**Title** Kaplan-Meier Estimator with Constraints for Right Censored Data  
-- a Recursive Computational Algorithm

**Version** 0.4-2

**Date** 2022-11-21

**Maintainer** Yifan Yang <yfyang.86@hotmail.com>

**Description** Given constraints for right censored data, we use a recursive computational algorithm to calculate the the ``constrained" Kaplan-Meier estimator. The constraint is assumed given in linear estimating equations or mean functions. We also illustrate how this leads to the empirical likelihood ratio test with right censored data and accelerated failure time model with given coefficients. EM algorithm from emplik package is used to get the initial value. The properties and performance of the EM algorithm is discussed in Mai Zhou and Yifan Yang (2015)<[doi:10.1007/s00180-015-0567-9](https://doi.org/10.1007/s00180-015-0567-9)> and Mai Zhou and Yifan Yang (2017) <[doi:10.1002/wics.1400](https://doi.org/10.1002/wics.1400)>. More applications could be found in Mai Zhou (2015) <[doi:10.1201/b18598](https://doi.org/10.1201/b18598)>.

**URL** <https://github.com/yfyang86/kmc/>

**License** LGPL-3

**Depends** R (>= 3.5), compiler, rootSolve, emplik

**Suggests** survival, ggplot2, tidyr, testthat (>= 3.0.0)

**LinkingTo** Rcpp

**NeedsCompilation** yes

**Encoding** UTF-8

**Config/testthat/edition** 3

**Author** Yifan Yang [aut, cre, cph],  
Mai Zhou [aut]

**Repository** CRAN

**Date/Publication** 2022-11-22 08:30:02 UTC

## Contents

check_G_mat . . . . .	2
kmc.bjtest . . . . .	3
kmc.clean . . . . .	5
kmc.solve . . . . .	6
plotkmc2D . . . . .	8
<b>Index</b>	<b>10</b>

---

check_G_mat	<i>Check the constraints of KMC</i>
-------------	-------------------------------------

---

### Description

To derive the empirical likelihood with constraints, we need to make sure there are solutions. Dines' method is used here to check whether the linear constraints are proper or not.

### Usage

```
check_G_mat(gmat)
```

### Arguments

gmat	A p by n. Here p is the number of constraints, n is the number of observations. The matrix is defined in <doi: 10.1201/b18598>.
------	---

### Value

flag	A flag: - 0: not proper - 1: proper
------	-------------------------------------

### Author(s)

Yifan Yang(yfyang.86@hotmail.com)

### References

- Dines, L. L. (1926). On positive solutions of a system of linear equations *Annals of Mathematics* pages 386–392
- Zhou, M. and Yang, Y. (2015). A recursive formula for the Kaplan-Meier estimator with mean constraints and its application to empirical likelihood *Computational Statistics*. **Online ISSN** 1613-9658.

**Examples**

```
#### A Proper Example ####
x <- c( 1, 1.5, 2, 3, 4.2, 5.0, 6.1, 5.3, 4.5, 0.9, 2.1, 4.3)
d <- c( 1, 1, 0, 1, 0, 1, 1, 1, 1, 0, 0, 1)
f1 <-function(x) { x - 3.7}
f2 <- function(x) {x^2 - 16.5 }
g <- list(f1, f2)
re = kmc.cclean(x, d)
p = length(g)
n = length(re$kmc.time)
gmat<-matrix(0, p, n);
for(i in 1:p){
  gmat[i,] = g[[i]](re$kmc.time)
}
# You may want to require(Rcpp) on some platforms (such Mac OSX-ARM)
# library(Rcpp)
# check_G_mat(gmat)
```

kmc.bjtest

*Calculate the NPMLE with constraints for accelerated failure time model with given coefficients.*

**Description**

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

$$El(F) = \prod_{i=1}^n (\Delta F(T_i))^{\delta_i} (1 - F(T_i))^{1-\delta_i}$$

with constraints

$$\sum_i g(T_i) \Delta F(T_i) = 0, \quad i = 1, 2, \dots$$

Instead of EM algorithm, this function calculates the Kaplan-Meier estimator with mean constraints recursively to test  $H_0 : \beta = \beta_0$  in the accelerated failure time model:

$$\log(T_i) = y_i = x_i \beta^\top + \epsilon_i,$$

where  $\epsilon$  is distribution free.

**Usage**

```
kmc.bjtest(y, d, x, beta, init.st="naive")
```

**Arguments**

y	Response variable vector (length n).
d	Status vector (length n), 0: right censored; 1 uncensored.
x	n by p explanatory variable matrix.
beta	The value of the regression coefficient vector (length p) to be tested.
init.st	Type of methods to initialize the algorithm. By default, init.st is set to 1/n

## Details

The empirical likelihood is the likelihood of the error term when the coefficients are specified. Model assumptions are the same as requirements of a standard Buckley-James estimator.

## Value

a list with the following components:

prob	the probabilities that max the empirical likelihood under estimating equation.
logel1	the log empirical likelihood without constraints, i.e. under Kaplan-Merier of residuals'
logel2	the log empirical likelihood with constraints, i.e. under null hypotheses or estimation equations.
"-2LLR"	the -2 loglikelihood ratio; have approximate chisq distribution under null hypotheses
convergence	an indicator: 0: fails to converge 1: converged

## Author(s)

Mai Zhou(mai@ms.uky.edu), Yifan Yang(yfyang.86@hotmail.com)

## References

- Buckley, J. and James, I. (1979). Linear regression with censored data. *Biometrika*, **66** 429-36
- Zhou, M., & Li, G. (2008). Empirical likelihood analysis of the Buckley-James estimator. *Journal of multivariate analysis*, **99**(4), 649-664.
- Zhou, M. and Yang, Y. (2015). A recursive formula for the Kaplan-Meier estimator with mean constraints and its application to empirical likelihood *Computational Statistics*. **Online ISSN** 1613-9658.

## See Also

[plotkmc2D](#), [bjtest](#).

## Examples

```
library(survival)
stanford5 <- stanford2[!is.na(stanford2$t5), ]
y <- log10(stanford5$time)
d <- stanford5$status
oy <- order(y, -d)
d <- d[oy]
y <- y[oy]
x <- cbind(1, stanford5$age)[oy,]
beta0 <- c(3.2, -0.015)
ss <- kmc.bjtest(y, d, x=x, beta=beta0, init.st="naive")
```

kmc.clean

*Perform Data Clean for the kmc Algorithm***Description**

The kmc.clean function clean the (kmc.time, delta) for the randomized censored data:

- Reorder the data according to the observed time and status;
- Clean the (right) censored data point(s) if they happen before the first uncensored data.
- If there are ties in the data. For the time points contain ties, e.g.

$$(T_{i_s}, d_{i_s}), i_s \in S \forall j \in S, T_j \equiv T$$

, we re-arranged the data in a manner that those with d=1 are ordered ahead of those with d=0. As d=0 indicates the data point is right censored, such procedure is trivial.

**Usage**

```
kmc.clean(kmc.time, delta)
```

**Arguments**

kmc.time	Non-negative real vector. The observed time.
delta	0/1 vector. Censoring status indicator, 0: right censored; 1 uncensored

**Value**

A list with the following components:

kmc.time	The cleaned observed time.
delta	The cleaned censoring status indicator, 0: right censored; 1 uncensored

**Author(s)**

Yifan Yang(yfyang.86@hotmail.com)

**References**

Zhou, M. and Yang, Y. (2015). A recursive formula for the Kaplan-Meier estimator with mean constraints and its application to empirical likelihood *Computational Statistics Online* ISSN 1613-9658.

**Examples**

```
x <- c( 1, 1.5, 2, 3, 4.2, 5.0, 6.1, 5.3, 4.5, 0.9, 2.1, 4.3)
d <- c( 1, 1, 0, 1, 0, 1, 1, 1, 1, 0, 0, 1)
kmc.clean(x, d)
```

kmc.solve

*Calculate NPMLE with constraints for right censored data***Description**

This function calculate the Kaplan-Meier estimator with mean constraints recursively.

$$El(F) = \prod_{i=1}^n (\Delta F(T_i))^{\delta_i} (1 - F(T_i))^{1-\delta_i}$$

with constraints

$$\sum_i g(T_i) \Delta F(T_i) = 0, \quad , i = 1, 2, \dots$$

It uses Lagrange multiplier directly.

**Usage**

```
kmc.solve(x, d, g, em.boost = T, using.num = T, using.Fortran =
          T, using.C = F, tmp.tag = T, rtol = 1e-09, control =
          list(nr.it = 20, nr.c = 1, em.it = 3),...)
```

**Arguments**

x	Non-negative real vector. The observed time.
d	0/1 vector. Censoring status indicator, 0: right censored; 1 uncensored
g	list of constraint functions. It should be a list of functions list(f1,f2,...)
em.boost	A logical value. It determines whether the EM algorithm is used to get the initial value, default=TRUE. See 'Details' for EM control.
using.num	A logical value. It determines whether the numeric derivatives is used in iterations, default=TRUE.
using.Fortran	A logical value. It determines whether Fortran is used in root solving, default=F.
using.C	A logical value. It determines whether to use Rcpp in iteraruib, default=T. This option will promote the computational efficiency of the KMC algorithm. Development version works on one constraint only, otherwise it will generate a Error information. It won't work on using.num=F.
tmp.tag	Development version needs it, keep it as TRUE.
rtol	Tolerance used in rootSolve(multiroot) package, see 'rootSolve::multiroot'.
control	A list. The entry nr.it controls max iterations allowed in N-R algorithm default=20; nr.c is the scaler used in N-R algorithm default=1; em.it is max iteration if use EM algorithm (em.boost) to get the initial value of lambda, default=3.
...	Unspecified yet.

**Details**

The function `check_G_mat` checks whether the solution space is null or not under the constraint. But due to the computational complexity, it will detect at most two conditions.

**Value**

A list with the following components:

<code>loglik.ha</code>	The log empirical likelihood without constraints
<code>loglik.h0</code>	The log empirical likelihood with constraints
<code>"-2LLR"</code>	The -2 Log empirical likelihood ratio
<code>phat</code>	$\Delta F(T_i)$
<code>pvalue</code>	The p-value of the test
<code>df</code>	Degree(s) of freedom. It equals the number of constraints.
<code>lambda</code>	The lambda is the Lagrangian multiplier described in reference.

**Author(s)**

Mai Zhou(mai@ms.uky.edu), Yifan Yang(yfyang.86@hotmail.com)

**References**

Zhou, M. and Yang, Y. (2015). A recursive formula for the Kaplan-Meier estimator with mean constraints and its application to empirical likelihood *Computational Statistics Online* ISSN 1613-9658.

**See Also**

[plotkmc2D](#).

**Examples**

```
# positive time
x <- c( 1, 1.5, 2, 3, 4.2, 5.0, 6.1, 5.3, 4.5, 0.9, 2.1, 4.3)
# status censored/uncensored
d <- c( 1, 1, 0, 1, 0, 1, 1, 1, 1, 0, 0, 1)

#####
# dim =1
#####

f <- function(x) {x-3.7}
g <- list(f=f) ;

kmc.solve(x, d, g) ;
kmc.solve(x, d, g, using.C=TRUE) ;

#####
```

```

# dim =2
#####

myfun5 <- function( x) {
  x^2-16.5
}

g <- list( f1=f,f2=myfun5) ;           #define constraint as a list

re0 <- kmc.solve(x,d,g);

#####
# Print Estimation and other information
# with option: digits = 5
#####

#' Print kmc object
#'
#' @param x kmc object
#' @param digits minimal number of significant digits, see print.default.
f_print <- function(x, digits = 5){
  cat("\n-----\n")
  cat("A Recursive Formula for the Kaplan-Meier Estimator with Constraint\n")
  cat("Information:\n")
  cat("Number of Constraints:\t", length(x$g), "\n")
  cat("lamda(s):\t", x$lambda,'\n');
  cat("\n-----\n")
  names <- c("Log-likelihood(Ha)", "Log-likelihood(H0)",
    "-2LLR", paste("p-Value(df=", length(x$g), ")", sep = ""))
  re <- matrix(c(x[[1]], x[[2]], x[[3]], 1 - pchisq(x[[3]],
    length(x$g))), nrow = 1)
  colnames(re) <- names
  rownames(re) <- "Est"
  print.default(format(re, digits = digits), print.gap = 2,
    quote = FALSE, df = length(x$g))
  cat("-----\n")
}

f_print(re0)

```

---

plotkmc2D

---

*Plot the contour plot of log-likelihood around the H0 (dim=2).*


---

### Description

Given a kmc object, this function will produce contour plot if there were two constraints.

### Usage

```

plotkmc2D(resultkmc, flist=list(f1=function(x){x}, f2=function(x){x^2}),
  range0=c(0.2, 3,20))

```

**Arguments**

resultkmc	S3 Object of kmcS3.
f1list	list of two functions, f1list=list( f1=function( x ) x ,f2=function( x ) x^2 )
range0	A vector that helps to determine the range of the contour plot, i.e (center[1]-range0[1], center[2]-range0[2]) to (center+range0[1], center[2]+range0[2]). The third parameter defines the number of grids would be used.

**Value**

X	x.grid
Y	y.grid
Z	grid value

**Author(s)**

Yifan Yang(yfyang.86@hotmail.com)

**Examples**

```
x <- c( 1, 1.5, 2, 3, 4.2, 5.0, 6.1, 5.3, 4.5, 0.9, 2.1, 4.3)
d <- c( 1, 1, 0, 1, 0, 1, 1, 1, 1, 0, 0, 1)

f<-function( x) { x-3.7}

myfun5 <- function( x) {
  x^2-16.5
}
# construct g as a LIST!

g=list( f1=f,f2=myfun5) ;
kmc.solve( x,d,g) ->re0;

#plotkmc2D(re0) ->ZZ; # run this to generate contour plot
#Advanced PLOT option using ggplot2: not run
#library(reshape2)
#volcano3d <- melt(ZZ$Z)
#names(volcano3d) <- c("x", "y", "z")

#volcano3d$x <- ZZ$x[volcano3d$x];
#volcano3d$y <- ZZ$y[volcano3d$y];

#### Plot: use ggplot2 ####
#library(ggplot2)
# v <- ggplot(volcano3d, aes(x, y, z=z));
# v + geom_tile(aes(fill = z)) +
#   stat_contour()+
#   scale_fill_gradientn("Custom Colours", colours=grey.colors(10));

#### Plot: use qplot ####
#qplot(x, y, z = z, data = volcano3d, stat = "contour", geom = "path")
```

# Index

`bjtest`, [4](#)

`check_G_mat`, [2](#)

`kmc.bjtest`, [3](#)

`kmc.clean`, [5](#)

`kmc.solve`, [6](#)

`plotkmc2D`, [4](#), [7](#), [8](#)