User guide for R package 'icemelt'

Soutrik Mandal
Department of Statistics, Texas A&M University
email: smandal@stat.tamu.edu

This package contains functions to estimate the regression parameters $\boldsymbol{\beta} = (\beta_1, \boldsymbol{\beta}_2^T)^T$ for the semi-parametric transformation model using naive, regression calibration, and the proposed imputation method (Mandal et al., 2018). Standard errors for these parameter estimates are also provided. The linear transformation model is given by

$$H(T) = -X\beta_1 - \boldsymbol{Z}^T\boldsymbol{\beta}_2 + e,$$

where T is the interval-censored time-to-event, H is an unknown monotone transformation function on $(0, \infty)$ with $H(T) \to -\infty$ as $T \to 0$, X is the covariate measured with error and $\boldsymbol{Z}$ is the error-free covariate, e is the error with a completely known distribution function and is independent of both X and $\boldsymbol{Z}$.

The observed data are collected from n independent subjects randomly drawn from an underlying population. The data from the ith subject is of the form $(L_i, R_i, \Delta_i, W_{i1}, \ldots, W_{im}, \boldsymbol{Z}_i)$, $i = 1, \ldots, n$. For the ith subject, if the right censoring indicator $\Delta_i = 1$, then the time-to-event T_i satisfies $L_i < T_i \leq R_i$, and when $\Delta_i = 0$, the subject is right censored above L_i, and $L_i < T_i < \infty$. Here $\boldsymbol{Z}_i$ is a $p \times 1$ vector of error free covariates, and $W_{i1}, \ldots, W_{im}$ denote the replicated measurements on a surrogate variable for X. The surrogacy implies that conditional on the true covariates, the surrogate variable is independent of the response T. We assume that the observed surrogate W is related to the unobserved covariate X through the additive measurement error model,

$$W_{ij} = X_i + U_{ij},\ j = 1, \ldots, m,$$

where measurement error U_{ij}'s are assumed to be independently and identically distributed (iid) Normal$(0, \sigma_u^2)$ variables. Furthermore, we assume that U is completely independent of other observed variables.

In the following example, we will simulate a dataset using the true parameter values $\beta_1 = -1$ and $\beta_2 = 1$ and then use the package to get the estimates of the parameters and their standard errors using the naive (NV), regression calibration (RC) and imputation (IM) methods.

```
## The following function is used in generating epsilon from its CDF. Here, a choice
## of r=0 generates Cox PH model and r=1 generates Proportional Odds model.
rsep= function(u,r)
{
if(r==0)  return( log(-log(1-u)) )
else  return( log((exp(-r*log(1-u))-1)/r) )
```

```
}

n= 200  # sample size
m= 10  # number of failure-time imputations
nrep= 2  # number of repeated measurements for W
gridlen= 0.1  # grid length to divide time intervals

result= NULL
set.seed(15)
# x= rnorm(n, mean= 0, sd=1)  # symmetric true X
x= (rgamma(n,shape=2,scale=2)-4)/sqrt(8)  # skewed true X
z2= rbinom(n,1,0.5)  # error-free covariate Z

## Generating the random error terms:
ugen= runif(n)
rfix= 0.0  characterizes the error-distribution
ep= rsep(ugen,rfix)

truebeta= c(-1,1)
logt= -truebeta[1]*x -truebeta[2]*z2 + ep + 3
ttime= exp(logt)  true failure time
cen= runif(n,0,0.0001)

## Creating the intervals (L_i, R_i], i = 1, ..., n
len= 0.12
taumat= matrix(0,n,10)
taumat[,10]= 9000000000  # a large value used to denote infinity for right-censored
# observations; if you're changing this, change rcpos below
taumat[,2]= cen
for(i1 in 3:9)
taumat[,i1]= taumat[,2]+(i1-1)*len
right1= rep(0,n)
left= rep(0,n)
for(i2 in 1:n)
{
lenleft= length(which(taumat[i2,2:9]<ttime[i2]))
leftpot= rep(0,8)
leftpot[1:lenleft]= 1
missvec1= c(rbinom(4,1,0.7),rbinom(4,1,0.5))
left[i2]= max(leftpot*missvec1*taumat[i2,2:9])
```

```
if(left[i2]==0)
left[i2]= taumat[i2,2]

lenright= length(which(taumat[i2,2:10]>ttime[i2]))
rightpot= rep(0,9)
rightpot[(9-lenright+1):9]= 1
missvec2= c(rbinom(4,1,0.7),rbinom(4,1,0.5),1)
right1temp= rightpot*missvec2*taumat[i2,2:10]
right1[i2]= min(right1temp[right1temp!=0])
}

## Identifying the positions of the right-censored observations
rcpos= which(right1==9000000000)
lrcpos= length(rcpos)
notrcpos= (1:n)[-rcpos]

deltatemp= rep(1,n)  # del=1 are uncensored observations
deltatemp[rcpos]= 0
k= sum(deltatemp)  # number of data points that are not right censored

## Generating measurement error from a skewed distribution
umat= matrix((rgamma(n*nrep,shape=2,scale=2)-4)*0.70711/sqrt(8),n,nrep)
wmat= x+umat

## Creating the data matrix and specifying the number of T and X imputations
datamat= cbind(left,right1,deltatemp,z2)
ntimp= m
nximp= 20

## Installing and loading the package in the R terminal
# install.packages("icemelt_1.0.zip")  # for windows
# install.packages("icemelt_1.0.tar.gz")  # for linux
library(icemelt)
```

The final step is feeding the arguments into the function:

- datamat: the datamatrix with columns 'left time', 'right time', 'right censoring indicator' and 'error-free covariate' strictly in that order
- wmat: the matrix of replicated surrogate measurements of the error-prone covariate

- rfix: a real number to specify a particular model from the general class of models included within the linear transformation models (for example: rfix=0 denotes Cox PH model and rfix=1 denotes Proportional Odds model)
- gridlen: distance between gridpoints in T imputation (smaller values indicate more grid points between the left and right ends of the interval)
- ntimp: number of T imputations
- nximp: number of X imputations

```
im(datamat, wmat, rfix, gridlen, ntimp, nximp)
```

The `im()` function returns the following estimates and standard errors:

```
IM estimate of beta_1 = −0.835492043971614
IM estimate of beta_2 = 0.301338376929423
Estimated standard error of beta_1 = 0.405580336044359
Estimated standard error of beta_2 = 0.557459001740593
```

Next we use the `rc()` and `nv()` functions to calculate the regression calibration and naive estimates respectively.

```
rc(datamat, wmat, rfix, gridlen, ntimp)
```

The `rc()` function returns the following estimates and standard errors:

```
RC estimate of beta_1 = −0.794495641757086
RC estimate of beta_2 = 0.329881716358324
Estimated standard error of beta_1 = 0.368754821525003
Estimated standard error of beta_2 = 0.551000977382565
```

```
nv(datamat, wmat, rfix, gridlen, ntimp)
```

The `nv()` function returns the following estimates and standard errors:

```
NV estimate of beta_1 = −0.680381434199727
NV estimate of beta_2 = 0.319433993348971
```

```
Estimated standard error of beta_1 = 0.316136715202299
Estimated standard error of beta_2 = 0.550850998188916
```

The functions `imw1()` and `rcw1()` are used when only one replication of the surrogate variable W is available. While everything else remain same as the example provided above, these functions require a known value for the measurement error variance because it is no longer possible to estimate it from only one replication. Note that the `nv()` function is capable of handling the single replication scenario. We use the same example as above after making these few modifications. The changed lines are highlighted.

```
## The following function is used in generating epsilon from its CDF. Here, a choice
## of r=0 generates Cox PH model and r=1 generates Proportional Odds model.
rsep= function(u,r)
{
if(r==0)  return( log(-log(1-u)) )
else  return( log((exp(-r*log(1-u))-1)/r) )
}

n= 200  # sample size
m= 10  # number of failure-time imputations
nrep= 1  # number of repeated measurements for W
sigma2u= 0.5  # measurement error variance
gridlen= 0.1  # grid length to divide time intervals

result= NULL
set.seed(15)
# x= rnorm(n, mean= 0, sd=1)  # symmetric true X
x= (rgamma(n,shape=2,scale=2)-4)/sqrt(8)  # skewed true X
z2= rbinom(n,1,0.5)  # error-free covariate Z

## Generating the random error terms:
ugen= runif(n)
rfix= 0.0  characterizes the error-distribution
ep= rsep(ugen,rfix)

truebeta= c(-1,1)
logt= -truebeta[1]*x -truebeta[2]*z2 + ep + 3
ttime= exp(logt)  true failure time
cen= runif(n,0,0.0001)

## Creating the intervals (L_i, R_i], i = 1, ..., n
len= 0.12
```

```
taumat= matrix(0,n,10)
taumat[,10]= 9000000000  # a large value used to denote infinity for right-censored
# observations; if you're changing this, change rcpos below
taumat[,2]= cen
for(i1 in 3:9)
taumat[,i1]= taumat[,2]+(i1-1)*len
right1= rep(0,n)
left= rep(0,n)
for(i2 in 1:n)
{
lenleft= length(which(taumat[i2,2:9]<ttime[i2]))
leftpot= rep(0,8)
leftpot[1:lenleft]= 1
missvec1= c(rbinom(4,1,0.7),rbinom(4,1,0.5))
left[i2]= max(leftpot*missvec1*taumat[i2,2:9])

if(left[i2]==0)
left[i2]= taumat[i2,2]

lenright= length(which(taumat[i2,2:10]>ttime[i2]))
rightpot= rep(0,9)
rightpot[(9-lenright+1):9]= 1
missvec2= c(rbinom(4,1,0.7),rbinom(4,1,0.5),1)
right1temp= rightpot*missvec2*taumat[i2,2:10]
right1[i2]= min(right1temp[right1temp!=0])
}

## Identifying the positions of the right-censored observations
rcpos= which(right1==9000000000)
lrcpos= length(rcpos)
notrcpos= (1:n)[-rcpos]

deltatemp= rep(1,n)  # del=1 are uncensored observations
deltatemp[rcpos]= 0
k= sum(deltatemp)  # number of data points that are not right censored

## Generating measurement error from a skewed distribution
umat= matrix((rgamma(n*nrep,shape=2,scale=2)-4)*0.70711/sqrt(8),n,nrep)
wmat= x+umat

## Creating the data matrix and specifying the number of T and X imputations
```

```
datamat= cbind(left,right1,deltatemp,z2)
ntimp= m
nximp= 20

## Installing and loading the package in the R terminal
# install.packages("icemelt_1.0.zip")  # for windows
# install.packages("icemelt_1.0.tar.gz")  # for linux
library(icemelt)
```

The final step is feeding the arguments into the function:

- datamat: the datamatrix with columns 'left time', 'right time', 'right censoring indicator' and 'error-free covariate' strictly in that order
- wmat: a matrix with only one column of replicated surrogate measurements of the error-prone covariate
- rfix: a real number to specify a particular model from the general class of models included within the linear transformation models (for example: rfix=0 denotes Cox PH model and rfix=1 denotes Proportional Odds model)
- gridlen: distance between gridpoints in T imputation (smaller values indicate more grid points between the left and right ends of the interval)
- ntimp: number of T imputations
- nximp: number of X imputations
- sigma2u: a known value for the measurement error variance

```
imw1(datamat, wmat, rfix, gridlen, ntimp, nximp, sigma2u)
```

The `imw1()` function returns the following estimates and standard errors:

```
IM estimate of beta_1 = −1.00497641099083
IM estimate of beta_2 = 0.311869864572144
Estimated standard error of beta_1 = 0.749604882264894
Estimated standard error of beta_2 = 0.566524912675682
```

Next we use the `rcw1()` and `nv()` functions to calculate the regression calibration and naive estimates

respectively.

```
rcw1(datamat, wmat, rfix, gridlen, ntimp, sigma2u)
```

The `rcw1()` function returns the following estimates and standard errors:

```
RC estimate of beta_1 = −0.723956835998469
RC estimate of beta_2 = 0.338662792549628
Estimated standard error of beta_1 = 0.390607561198743
Estimated standard error of beta_2 = 0.550054626154455
```

```
nv(datamat, wmat, rfix, gridlen, ntimp)
```

The `nv()` function returns the following estimates and standard errors:

```
NV estimate of beta_1 = −0.508612125274339
NV estimate of beta_2 = 0.341956310319969
Estimated standard error of beta_1 = 0.273437252219733
Estimated standard error of beta_2 = 0.548524918869787
```

Reference

Mandal, S., Wang, S., and Sinha, S. (2018). Analysis of linear transformation models with covariate measurement error and interval censoring.