This manual illustrates how to use the R package 'mesub'

This packages estimates the regression parameters $\boldsymbol{\beta} = (\boldsymbol{\beta}_1^T, \beta_2)^T$ of the model

$$H(T) = -\boldsymbol{\beta}_1^{\mathrm{T}} \boldsymbol{Z} - \beta_2 X + e,$$

when X is not observed in the sample data. Here H is an unknown monotone transformation function, e is a random variable with a known distribution and is independent of $\boldsymbol{Z}$ and X.

The observed data are n iid copies of $(T^*, \Delta, \boldsymbol{Z}, \boldsymbol{W}^*)$, where $\boldsymbol{Z}$ is a $p \times 1$ vector of observed covariates, $T^* = \min(T, C)$, where T is the time-to-failure, C is the censoring time, and $\Delta = I(T \leq C)$. We assume that conditional on the complete covariates $(X, \boldsymbol{Z})$, where X is unobservable, T and C are independent. Here $\boldsymbol{W}^* = (W_1^*, \cdots, W_m^*)^{\mathrm{T}}$ is a surrogate measurement for the scalar covariate X. We assume that

$$W_{ij}^* = X_i + U_{ij}^*,\ i = 1, \cdots, n,\ j = 1, \cdots, m,$$

where the measurement errors U_{ij}^* are assumed to be iid copies of a random variable U^* which follows a symmetric distribution centered at 0. We assume that U^* is independent of $(T, X, \boldsymbol{Z}, C)$.

In the following examples we will simulate data using varying mechanism, and then use the package to get the estimates of the parameters and the standard errors.

```
library(mesub)
set.seed(1)
r=0
m=2
trbeta1=1
trbeta2=1
sigmau=0.71
n=200 #Sample size
ran=runif(n, 0, 1)
nby3=as.integer(0.33*n)
x=c(rnorm(nby3, -0.6, 0.5), rnorm((n-nby3), 1.25, 0.5)) # true covariate x
x=x+6
z=runif(n, 0, 1)  # error free covariate
if(r>0) epsilon=log(( (1+0)*exp(-r*log(ran))   -1)/r) else epsilon=log(-log(ran))
t=exp(-trbeta1*z-trbeta2*x+epsilon) # this is the  actual time to event
cns=runif(n, 0, 6) # 10% censored data with r=0
delta=as.numeric(t<=cns) # right censoring indicator
tstar=apply(cbind(t, cns), 1, min)
ustar=matrix(rnorm((m*n), 0, sigmau), ncol=m, nrow=n) # measurement error
wstar=x+ustar
wstar=wstar/sd(wstar)  # It is aways better to use rescaled values
z=as.matrix(z)
## End of data generation
out=ltm.me2(tstar, delta, wstar, z, r, 2)
out
$naive.estimate
[1] 1.125509 1.023513
# here 1.125509 is the estimate of $\beta_1$ and 1.023513
# is the estimate for $\beta_2$
```

```
$naive.vcov
            [,1]        [,2]
[1,] 0.071864708 0.004930186
[2,] 0.004930186 0.009131966
$naive.se
[1] 0.26807594 0.09556132
$our.estimate
[1] 1.444724 1.695173
$our.vcov
           [,1]       [,2]
[1,] 0.11731595 0.02277929
[2,] 0.02277929 0.06662100
$our.se
[1] 0.3425142 0.2581104
$AIC
[1] 151.9277
$BIC
[1] 175.0159
out=ltm.me2(tstar, delta, wstar, z, r, 1)
> out
$naive.estimate
[1] 1.125509 1.023513
$naive.vcov
            [,1]        [,2]
[1,] 0.071864708 0.004930186
[2,] 0.004930186 0.009131966
$naive.se
[1] 0.26807594 0.09556132
$our.estimate
[1] 1.405962 1.708010
$our.vcov
           [,1]       [,2]
[1,] 0.13117580 0.03455378
[2,] 0.03455378 0.07227576
$our.se
[1] 0.3621820 0.2688415
$AIC
[1] 163.8657
$BIC
[1] 173.7606
```

The above results clearly indicate that the estimates due to the two methods are quite different. Furthermore, based on the BIC critera we should prefer a normal model for X given Z over a mixture model for X given Z. In the next example we simulate measurement errors from an uniform distribution.

```
library(mesub)
set.seed(1)
r=0
m=3
trbeta1=1
trbeta2=1
sigmau=0.71
```

```
 n=200 #Sample size
 ran=runif(n, 0, 1)
 nby3=as.integer(0.33*n)
 x=c(rnorm(nby3, -0.6, 0.5), rnorm((n-nby3), 1.25, 0.5)) # true covariate x
 z=runif(n, 0, 1)  # error free covariate
 if(r>0) epsilon=log(( (1+0)*exp(-r*log(ran))   -1)/r) else epsilon=log(-log(ran))
 t=exp(-trbeta1*z-trbeta2*x+epsilon) # this is the  actual time to event
 cns=runif(n, 0, 6) # 10% censored data with r=0
 delta=as.numeric(t<=cns) # right censoring indicator
 tstar=apply(cbind(t, cns), 1, min)
 ustar=matrix(runif((m*n), -1.75, 1.75)*sigmau, ncol=m, nrow=n) # measurement error
 wstar=x+ustar
 wstar=wstar  # It is aways better to use rescaled values
 z=as.matrix(z)
 ptime=proc.time()
 out=ltm.me2(tstar, delta, wstar, z, r, 2)
 ptime=proc.time()-ptime
 ptime
   user  system elapsed
 23.279   0.001  25.048
 out
$naive.estimate
[1] 1.2392133 0.8885428
$naive.vcov
            [,1]        [,2]
[1,] 0.077681536 0.005833456
[2,] 0.005833456 0.008015719
$naive.se
[1] 0.27871408 0.08953055
$our.estimate
[1] 1.391408 1.166867
$our.vcov
           [,1]       [,2]
[1,] 0.11180842 0.01634633
[2,] 0.01634633 0.02605338
$our.se
[1] 0.3343777 0.1614106
$AIC
[1] 229.4864
$BIC
[1] 252.5746

> out=ltm.me2(tstar, delta, wstar, z, r, 1)
 out
$naive.estimate
[1] 1.2392133 0.8885428
$naive.vcov
            [,1]        [,2]
[1,] 0.077681536 0.005833456
[2,] 0.005833456 0.008015719
$naive.se
[1] 0.27871408 0.08953055
$our.estimate
[1] 1.424818 1.168084
```

```
$our.vcov
           [,1]       [,2]
[1,] 0.11832461 0.02040846
[2,] 0.02040846 0.02730206
$our.se
[1] 0.3439834 0.1652333
$AIC
[1] 245.47
$BIC
[1] 255.365
```

The following example considers a linear transformation model with $r = 2$ and with four covariates.

```
set.seed(1)
 r=2
 m=4
 trbeta1=0.5
 trbeta2=0.5
 trbeta3=0.35
 trbeta4=0.85
 sigmau=1.0
 n=500 #Sample size
 ran=runif(n, 0, 1)
 nby3=as.integer(0.33*n)
 x=c(rnorm(nby3, -0.6, 0.5), rnorm((n-nby3), 1.25, 0.5)) # true covariate x
 z1=runif(n, 0, 1)  # error free covariate
 z2=rnorm(n, 0, 1)  # error free covariate
 z3=rbinom(n, 1, 0.28)# error free covariate
 if(r>0) epsilon=log(( (1+0)*exp(-r*log(ran))   -1)/r) else epsilon=log(-log(ran))
 t=exp(-trbeta1*z1-trbeta2*z2-trbeta3*z3-trbeta4*x+epsilon) # this
 # is the  actual time to event
 cns=runif(n, 0, 6) # 10% censored data with r=0
 delta=as.numeric(t<=cns) # right censoring indicator
 tstar=apply(cbind(t, cns), 1, min)
 ustar=matrix(runif((m*n), -1.75, 1.75)*sigmau, ncol=m, nrow=n) # measurement error
 wstar=x+ustar
 wstar=wstar  # It is aways better to use rescaled values
 z=cbind(z1, z2, z3)
 z=as.matrix(z)
 ptime=proc.time()
out=ltm.me2(tstar, delta, wstar, z, r, 2)
ptime= proc.time()-ptime
ptime
 user  system elapsed
262.752   0.013 262.963
 out
$naive.estimate
[1] 0.6437162 0.3406450 0.4619951 0.6426618
# the first three components of naive.estimate  are for z1, z2, z3,
# respectively, and 0.6426618 corresponds to x
$naive.vcov
            [,1]          [,2]         [,3]          [,4]
[1,] 0.157511775  2.504083e-03 0.0010257574  2.297990e-03
```

```
[2,] 0.002504083  1.420488e-02 0.0007483502 -6.941956e-06
[3,] 0.001025757  7.483502e-04 0.0671921231  8.035732e-04
[4,] 0.002297990 -6.941956e-06 0.0008035732  1.242336e-02
$naive.se
[1] 0.3968775 0.1191842 0.2592144 0.1114601
$our.estimate
[1] 0.6404138 0.3351174 0.4636812 0.8453742
$our.vcov
             [,1]          [,2]         [,3]          [,4]
[1,] 0.1621392059  2.477169e-03 0.0006852633  2.877967e-03
[2,] 0.0024771690  1.471947e-02 0.0006702920 -5.460864e-06
[3,] 0.0006852633  6.702920e-04 0.0693538997  9.261138e-04
[4,] 0.0028779674 -5.460864e-06 0.0009261138  2.324422e-02
$our.se
[1] 0.4026651 0.1213238 0.2633513 0.1524606
$AIC
[1] 618.5021
$BIC
[1] 664.8628
 out=ltm.me2(tstar, delta, wstar, z, r, 1)
 out
$naive.estimate
[1] 0.6437162 0.3406450 0.4619951 0.6426618
$naive.vcov
            [,1]          [,2]         [,3]          [,4]
[1,] 0.157511775  2.504083e-03 0.0010257574  2.297990e-03
[2,] 0.002504083  1.420488e-02 0.0007483502 -6.941956e-06
[3,] 0.001025757  7.483502e-04 0.0671921231  8.035732e-04
[4,] 0.002297990 -6.941956e-06 0.0008035732  1.242336e-02
$naive.se
[1] 0.3968775 0.1191842 0.2592144 0.1114601
$our.estimate
[1] 0.6627085 0.3373030 0.4706584 0.8262940
$our.vcov
             [,1]         [,2]         [,3]         [,4]
[1,] 0.1620212875 2.427448e-03 0.0007725748 3.328742e-03
[2,] 0.0024274481 1.468441e-02 0.0006527443 4.349151e-05
[3,] 0.0007725748 6.527443e-04 0.0691885159 1.138072e-03
[4,] 0.0033287422 4.349151e-05 0.0011380721 2.215119e-02
$our.se
[1] 0.4025187 0.1211793 0.2630371 0.1488327
$AIC
[1] 613.636
$BIC
[1] 634.709
```