Example MultiCure Code

Lauren J Beesley

Department of Biostatistics, University of Michigan

Contact: lbeesley@umich.edu

September 14, 2017

In this vignette, we provide code for fitting multistate cure models using the package *MultiCure* for an example dataset.

Load in Code

```
library(devtools)
install_github("lbeesleyBIOSTAT/MultiCure")
library(MultiCure)
```

View Vignettes

```
browseVignettes('MultiCure')
```

Simulate Data

The following code simulates datasets under a multistate cure model with 1) no covariate missingness or unequal censoring, 2) covariate missingness, and 3) unequal censoring

```
NONE = SimulateMultiCure(type = 'NoMissingness')
COV = SimulateMultiCure(type = 'CovariateMissingness')
CENS = SimulateMultiCure(type = 'UnequalCensoring')
```

Visualize Data

The following is a RShiny function for visualizing the observed information for recurrence and death. We note that there is not unequal censoring in this simulated dataset.Note: We have a very small probability of experiencing a recurrence after about time 50 (although they may rarely happen), so it seems generally reasonable to assume subjects still at risk for recurrence and death after time 50 are cured.

Data with no unequal censoring:

```
datWIDE = data.frame( Y_R = NONE$Y_R, Y_D = NONE$Y_D, delta_R = NONE$delta_R,
   delta_D = NONE$delta_D)
VISUALIZEDATA(datWIDE)
```

Data with unequal censoring:

```
datWIDE = data.frame( Y_R = CENS$Y_R, Y_D = CENS$Y_D, delta_R = CENS$delta_R,
   delta_D = CENS$delta_D)
VISUALIZEDATA(datWIDE)
```

Fitting to Complete Data with Weibull Baselines

```
Cov = data.frame(X1 = NONE$X1,X2 = NONE$X2)
VARS = names(Cov)
TransCov = list(Trans13 = VARS, Trans24 = VARS, Trans14 = VARS, Trans34 = VARS
    , PNonCure = VARS)
datWIDE = data.frame( Y_R = NONE$Y_R, Y_D = NONE$Y_D, delta_R = NONE$delta_R ,
     delta_D = NONE$delta_D, G = NONE$G)
```

Parameter Estimation:

```
fit = MultiCure(iternum = 50, datWIDE  = datWIDE, Cov = Cov, trace = T, ASSUME
    = 'SameHazard', TransCov = TransCov, BASELINE = 'weib')
beta = fit[[1]]
alpha = fit[[2]]
scale = fit[[3]]
shape = fit[[4]]
beta_save = fit[[5]]
alpha_save = fit[[6]]
scale_save = fit[[7]]
shape_save = fit[[8]]
```

Variance Estimation:

```
OUT = VarianceEM(fit,iternum=20, bootnum=50, datWIDE, Cov, ASSUME = '
    SameHazard', TransCov = TransCov, BASELINE = 'weib')
```

Fitting to Data with Covariate Missingness with Weibull Baselines

```
Cov = data.frame(X1 = COV$X1, X2 = COV$X2)
VARS = names(Cov)
TransCov = list(Trans13 = VARS, Trans24 = VARS, Trans14 = VARS, Trans34 = VARS
   , PNonCure = VARS)
datWIDE = data.frame( Y_R = COV$Y_R, Y_D = COV$Y_D, delta_R = COV$delta_R ,
   delta_D = COV$delta_D, G = COV$G)
```

Parameter Estimation:

```
ITERNUM = 200
fit = MultiCure(iternum = ITERNUM, datWIDE  = datWIDE, Cov = Cov,
   COVIMPUTEFUNCTION  = COVIMPUTEFUNCTION, COVIMPUTEINITIALIZE =
   COVIMPUTEINITIALIZE, IMPNUM = 10, trace = T, ASSUME = 'SameHazard',
   TransCov = TransCov, BASELINE = 'weib')
beta_save = fit[[5]]
alpha_save = fit[[6]]
scale_save = fit[[7]]
shape_save = fit[[8]]
### Theta-Hat can be estimated as the mean of the last few iterations
beta = apply(beta_save[,(ITERNUM-10): ITERNUM], 1, mean)
alpha =  apply(alpha_save[,(ITERNUM-10): ITERNUM], 1, mean)
scale =  apply(scale_save[,(ITERNUM-10): ITERNUM], 1, mean)
shape =  apply(shape_save[,(ITERNUM-10): ITERNUM], 1, mean)
```

Proper Imputations:

```
Proper = ProperDraws_MC(datWIDE,Cov, CovImp = fit[[9]], GImp = fit[[10]],
   YRImp = fit[[11]], deltaRImp = fit[[12]], COVIMPUTEFUNCTION =
   COVIMPUTEFUNCTION,  COVIMPUTEINITIALIZE = COVIMPUTEINITIALIZE, ASSUME = '
   SameHazard', TransCov = TransCov, BASELINE = 'weib')
CovImp = Proper[[1]]
GImp = Proper[[2]]
YRImp = Proper[[3]]
deltaRImp = Proper[[4]]
```

Variance Estimation Method 1: This code performs variance estimation by fitting a multistate cure model to each one of the imputed datasets. For each imputed dataset, we use the standard software estimated SEs from the Logistic regression and proportional hazards model fits. Then, Rubin's rules are applied to obtain the final variance estimator across imputed datasets. This function also provides a different estimate of Theta based on Rubin's rules (based only on the final iteration). We have found the estimate from the mean of the last few iterations (above) may have slightly better coverage unless the number of imputations is large.

```
OUT = VarianceMCEM_NOBOOT(fit,datWIDE,  CovImp, GImp, YRImp, deltaRImp, ASSUME
    = 'SameHazard', TransCov = TransCov, BASELINE = 'weib')
```

Variance Estimation Method 2: This code performs variance estimation by fitting a multistate cure model to each one of the imputed datasets. For each imputed dataset, variances are estimated by fitting the multistate cure model to many bootstrap samples the imputed data. Then Rubin's rules are applied to obtain the final variance estimator across

imputed datasets. This function also provides a different estimate of Theta based on Rubin's rules.

```
OUT = VarianceMCEM_BOOT(fit,bootnum = 50, datWIDE = datWIDE,  CovImp, GImp,
    YRImp, deltaRImp, ASSUME = 'SameHazard', TransCov = TransCov, BASELINE = '
    weib')
```

Fitting to Data with Unequal Censoring with Weibull Baselines

```
Cov = data.frame(X1 = CENS$X1, X2 = CENS$X2)
VARS = names(Cov)
TransCov = list(Trans13 = VARS, Trans24 = VARS, Trans14 = VARS, Trans34 = VARS
   , PNonCure = VARS)
datWIDE = data.frame( Y_R = CENS$Y_R, Y_D = CENS$Y_D, delta_R = CENS$delta_R,
   delta_D = CENS$delta_D, G = CENS$G)
```

Parameter Estimation:

```
ITERNUM = 200
fit = MultiCure(iternum = ITERNUM, datWIDE  = datWIDE, Cov = Cov, IMPNUM = 10,
   trace = T, ASSUME = 'SameHazard', TransCov = TransCov, BASELINE = 'weib',
   UNEQUALCENSIMPUTE = UNEQUALCENSIMPUTEWEIBREJECTION)
beta_save = fit[[5]]
alpha_save = fit[[6]]
scale_save = fit[[7]]
shape_save = fit[[8]]
### Theta-Hat can be estimating as the mean of the last few iterations
beta = apply(beta_save[,(ITERNUM-10): ITERNUM], 1, mean)
alpha =  apply(alpha_save[,(ITERNUM-10): ITERNUM], 1, mean)
scale =  apply(scale_save[,(ITERNUM-10): ITERNUM], 1, mean)
shape =  apply(shape_save[,(ITERNUM-10): ITERNUM], 1, mean)
```

Proper Imputations:

```
Proper = ProperDraws_MC(datWIDE,Cov, CovImp = fit[[9]], GImp = fit[[10]],
   YRImp = fit[[11]], deltaRImp = fit[[12]], ASSUME = 'SameHazard', TransCov
   = TransCov, BASELINE = 'weib', UNEQUALCENSIMPUTE =
   UNEQUALCENSIMPUTEWEIBREJECTION)
CovImp = Proper[[1]]
GImp = Proper[[2]]
YRImp = Proper[[3]]
deltaRImp = Proper[[4]]
```

Variance Estimation Method 1:

```
OUT = VarianceMCEM_NOBOOT(fit,datWIDE,  CovImp, GImp, YRImp, deltaRImp, ASSUME
    = 'SameHazard', TransCov = TransCov, BASELINE = 'weib')
```

Fitting to Complete Data with Cox Baselines

```
Cov = data.frame(X1 = NONE$X1,X2 = NONE$X2)
VARS = names(Cov)
TransCov = list(Trans13 = VARS, Trans24 = VARS, Trans14 = VARS, Trans34 = VARS
    , PNonCure = VARS)
datWIDE = data.frame( Y_R = NONE$Y_R, Y_D = NONE$Y_D, delta_R = NONE$delta_R ,
     delta_D = NONE$delta_D, G = NONE$G)
```

Parameter Estimation:

```
fit = MultiCure(iternum = 50, datWIDE  = datWIDE, Cov = Cov, trace = T, ASSUME
    = 'SameHazard', TransCov = TransCov, BASELINE = 'cox')
beta = fit[[1]]
alpha = fit[[2]]
beta_save = fit[[3]]
alpha_save = fit[[4]]
```

Variance Estimation:

```
OUT = VarianceEM(fit,iternum=20, bootnum=50, datWIDE, Cov, ASSUME = '
    SameHazard', TransCov = TransCov, BASELINE = 'cox')
```

Fitting to Data with Covariate Missingness with Cox Baselines

```
Cov = data.frame(X1 = COV$X1, X2 = COV$X2)
VARS = names(Cov)
TransCov = list(Trans13 = VARS, Trans24 = VARS, Trans14 = VARS, Trans34 = VARS
   , PNonCure = VARS)
datWIDE = data.frame( Y_R = COV$Y_R, Y_D = COV$Y_D, delta_R = COV$delta_R ,
   delta_D = COV$delta_D, G = COV$G)
```

Parameter Estimation:

```
ITERNUM = 200
fit = MultiCure(iternum = ITERNUM, datWIDE  = datWIDE, Cov = Cov,
   COVIMPUTEFUNCTION  = COVIMPUTEFUNCTION, COVIMPUTEINITIALIZE =
   COVIMPUTEINITIALIZE, IMPNUM = 10,trace = T, ASSUME = 'SameHazard',
   TransCov = TransCov, BASELINE = 'cox')
beta_save = fit[[3]]
alpha_save = fit[[4]]
### Theta-Hat can be estimated as the mean of the last few iterations
beta = apply(beta_save[,(ITERNUM-10): ITERNUM], 1, mean)
alpha =  apply(alpha_save[,(ITERNUM-10): ITERNUM], 1, mean)
```

Proper Imputations:

```
Proper = ProperDraws_MC(datWIDE,Cov, CovImp = fit[[5]], GImp = fit[[6]], YRImp
    = fit[[7]], deltaRImp = fit[[8]], COVIMPUTEFUNCTION = COVIMPUTEFUNCTION,
    COVIMPUTEINITIALIZE = COVIMPUTEINITIALIZE, ASSUME = 'SameHazard',
   TransCov = TransCov, BASELINE = 'cox')
CovImp = Proper[[1]]
GImp = Proper[[2]]
YRImp = Proper[[3]]
deltaRImp = Proper[[4]]
```

Variance Estimation Method 1:

```
OUT = VarianceMCEM_NOBOOT(fit,datWIDE,  CovImp, GImp, YRImp, deltaRImp, ASSUME
    = 'SameHazard', TransCov = TransCov, BASELINE = 'cox')
```

Variance Estimation Method 2:

```
OUT = VarianceMCEM_BOOT(fit,bootnum = 50, datWIDE = datWIDE,  CovImp, GImp,
   YRImp, deltaRImp, ASSUME = 'SameHazard', TransCov = TransCov, BASELINE = '
   cox')
```

Fitting to Data with Unequal Censoring with Cox Baselines

```
Cov = data.frame(X1 = CENS$X1, X2 = CENS$X2)
VARS = names(Cov)
TransCov = list(Trans13 = VARS, Trans24 = VARS, Trans14 = VARS, Trans34 = VARS
   , PNonCure = VARS)
datWIDE = data.frame( Y_R = CENS$Y_R, Y_D = CENS$Y_D, delta_R = CENS$delta_R,
   delta_D = CENS$delta_D, G = CENS$G)
```

Parameter Estimation:

```
ITERNUM = 200
fit = MultiCure(iternum = ITERNUM, datWIDE  = datWIDE, Cov = Cov, IMPNUM = 10,
   trace = T, ASSUME = 'SameHazard', TransCov = TransCov, BASELINE = 'cox',
   UNEQUALCENSIMPUTE = UNEQUALCENSIMPUTECOXMH)
beta_save = fit[[3]]
alpha_save = fit[[4]]
### Theta-Hat can be estimating as the mean of the last few iterations
beta = apply(beta_save[,(ITERNUM-10): ITERNUM], 1, mean)
alpha =  apply(alpha_save[,(ITERNUM-10): ITERNUM], 1, mean)
```

Proper Imputations:

```
Proper = ProperDraws_MC(datWIDE,Cov, CovImp = fit[[5]], GImp = fit[[6]], YRImp
    = fit[[7]], deltaRImp = fit[[8]], ASSUME = 'SameHazard', TransCov =
   TransCov, BASELINE = 'cox', UNEQUALCENSIMPUTE = UNEQUALCENSIMPUTECOXMH)
CovImp = Proper[[1]]
GImp = Proper[[2]]
YRImp = Proper[[3]]
deltaRImp = Proper[[4]]
```

Variance Estimation Method 1:

```
OUT = VarianceMCEM_NOBOOT(fit,datWIDE,  CovImp, GImp, YRImp, deltaRImp, ASSUME
    = 'SameHazard', TransCov = TransCov, BASELINE = 'cox')
```

Examine Convergence of the Parameter Values

```
cbPalette <- c("#999999", "#E69F00", "#56B4E9", "#009E73", "#F0E442", "#0072B2
    ", "#D55E00", "#CC79A7")

par(mfrow=c(1,3))

matplot(t(beta_save[c(1,3,5,7),]), type = 'l',lty = 1, lwd = 2, col =
    cbPalette[c(6,7,4,8)], main = 'Beta Parameters for X1', xlab = 'Iterations
    ', ylab = 'Estimated Value')
legend(x='topright', fill = cbPalette[c(6,7,4,8)], legend = c( 'Transition
    1->3', 'Transition 2->4', 'Transition 1->4', 'Transition 3->4'), cex =
    0.8)

matplot(t(beta_save[c(2,4,6,8),]), type = 'l',lty = 1, lwd = 2, col =
    cbPalette[c(6,7,4,8)], main = 'Beta Parameters for X2', xlab = 'Iterations
    ', ylab = 'Estimated Value')
legend(x='bottomright', fill = cbPalette[c(6,7,4,8)], legend = c( 'Transition
    1->3', 'Transition 2->4', 'Transition 1->4', 'Transition 3->4'), cex =
    0.8)

matplot(t(alpha_save), type = 'l',lty = 1, lwd = 2, col =  cbPalette[c(6,7,4)
    ], main = 'Alpha Parameters', xlab = 'Iterations', ylab = 'Estimated Value
    ')
legend(x='topright', fill = cbPalette[c(6,7,4,8)], legend = c( 'Intercept', '
    X1', 'X2'), cex = 0.8)
```

Estimate State Occupancy Probabilities

Weibull Baseline Hazards:

```
STATEOCCUPANCYWEIB(times = seq(0,max(datWIDE$Y_D),1), TransCov, newCov = data.
   frame(X1 = c(0,0.5), X2 = c(0,0.5)), beta, alpha, scale, shape)
```

Cox Baseline Hazards, EM Algorithm:

```
Haz = BaselineHazard_NOIMP(datWIDE, Cov, beta, alpha, TransCov, ASSUME = '
   SameHazard', p=p_save[,length(p_save[1,])])
STATEOCCUPANCYCOX_NOIMP(times = seq(0,max(datWIDE$Y_D),1), TransCov, newCov =
   data.frame(X1 = c(0,0.5), X2 = c(0,0.5)), beta, alpha, Haz_13 = Haz[[1]],
   Haz_24 = Haz[[2]], Haz_14 = Haz[[3]], Haz_34 = Haz[[4]])
```

Cox Baseline Hazards, MCEM Algorithm: (Be Patient, the integrations are tricky)

```
haz = Baselinehazard_IMP(datWIDE, CovImp,GImp, YRImp,deltaRImp, beta, alpha,
   TransCov, ASSUME = 'SameHazard')
STATEOCCUPANCYCOX_IMP(times = seq(0,max(datWIDE$Y_D),5), TransCov, newCov =
   data.frame(X1 = c(0), X2 = c(0)), beta, alpha, Basehaz13 = haz[[1]],
   Basehaz24 = haz[[2]], Basehaz14 = haz[[3]], Basehaz34 = haz[[4]])
```