Using Package MultiCure to Fit Multistate Cure Models

Lauren J Beesley

Department of Biostatistics, University of Michigan

Contact: lbeesley@umich.edu

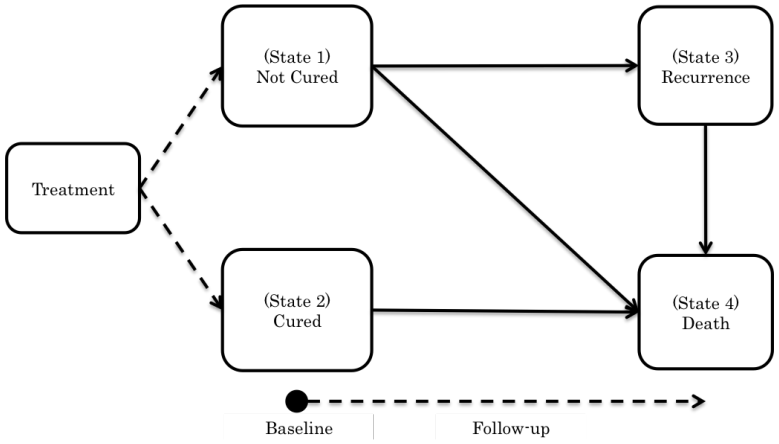
September 13, 2017

In this vignette, we provide a brief introduction to using the R package *MultiCure* to fit multistate cure models. We will not include technical details about the multistate cure model or the model fitting algorithms here. We will generally note that this R package includes two different model fitting algorithms: the EM algorithm and the Monte Carlo EM algorithm. The EM algorithm is used for fitting the multistate cure model when we have missingness in partially latent cure status but have no other sources of missingness. The Monte Carlo EM algorithm was developed to handle two additional sources of missingness: missingness in the covariates and missingness in the outcome information related to having unequal follow-up for the two outcome events. This latter form of missingness is common in the setting where the outcomes of interest are recurrence and death.

1 Multistate Cure Model Structure

For details about the multistate cure model structure, see Conlon *and others* (2013). Here, we will just list basic details. **Figure 1** shows the structure of the multistate cure model used in this package. We note that the outcome events may be events other than recurrence and death, but we will use these events as examples.

Figure 1: KM Plots for Time to Recurrence and Death in Illustrative Data



We model the probability of not being cured by initial treatment using logistic regression:

$$\text{logit}(P(\text{ Not Cured } = 1|\text{ Covariates } X_i)) = \alpha_0 + \alpha_1^T X_i$$

We model the transition rate from state j to state k for all transitions except $3 \to 4$ using proportional hazards model:

$$\lambda_{jk}(t) = \lambda_{jk}^0(t) \exp(\beta_{jk}^T X_i)$$

In the model for the $3 \to 4$ transition, we use the "clock reset" method in which time is reset to zero upon entering state 3, and use a proportional hazards regression to model the residual time in state 3 before entering state 4 as follows:

$$\lambda_{34}(t - T_{ir}) = \mathbb{I}(t > T_{ir})\lambda_{34}^0(t - T_{ir}) \exp(\beta_{34}^T X_i)$$

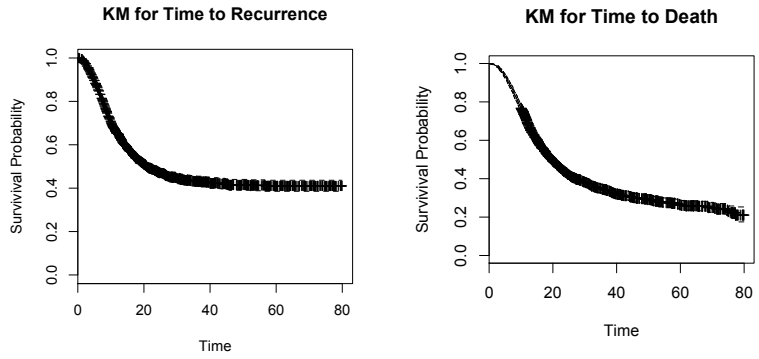
where T_{ir} is the time of recurrence. We can incorporate the time spent in state 1, T_{ir}, as a covariate in X_i for this transition if desired. We may use parametric or nonparametric forms for the baseline hazards.

Let $\Lambda_{jk}(t)$ and $\Lambda_{jk}^0(t)$ represent the cumulative hazard and cumulative baseline hazard for transition $j \to k$. We consider several possible restrictions on the baseline hazards for the $2 \to 4$ and $1 \to 4$ transitions, which may typically represent death from other causes. It may be reasonable to assume that the hazards are identical ($\Lambda_{14}(t) = \Lambda_{24}(t)\ \forall\ t \geq 0$). In this case, the multistate cure model reduces to a CPH cure model for recurrence time with two additional regressions for time to death with and without recurrence (Conlon *and others*, 2013). We may also assume the hazards are proportional ($\Lambda_{14}(t) = \Lambda_{24}(t) \exp\{\beta_0\}$) or that the baseline hazards are equal ($\Lambda_{14}^0(t) = \Lambda_{24}^0(t)$, β_{24} and β_{14} unrestricted), or we may make no equality assumptions ($\Lambda_{14}(t)$ and $\Lambda_{24}(t)$ unrestricted).

2 Model Fitting Software

We simulate a single dataset under a multistate cure model with two multivariate normal covariates. **Figure 2** shows plots resulting from applying the Kaplan-Meier estimator to time to recurrence and time to death for this example dataset.

Figure 2: KM Plots for Time to Recurrence and Death in Illustrative Data



We note that, since recurrence and death from other causes are competing events, the Kaplan-Meier estimate for time to recurrence does not represent the true recurrence rate

in the population. However, we can use this Kaplan-Meier plot to evaluate when during follow-up the recurrences are occurring. In order for the proposed methods to perform well, we need to have sufficient follow-up for both recurrence and death. In particular, the Kaplan-Meier plot for recurrence needs to plateau, and we may suppose that subjects at risk after some late follow-up time (eg. 50 in this example) are cured.

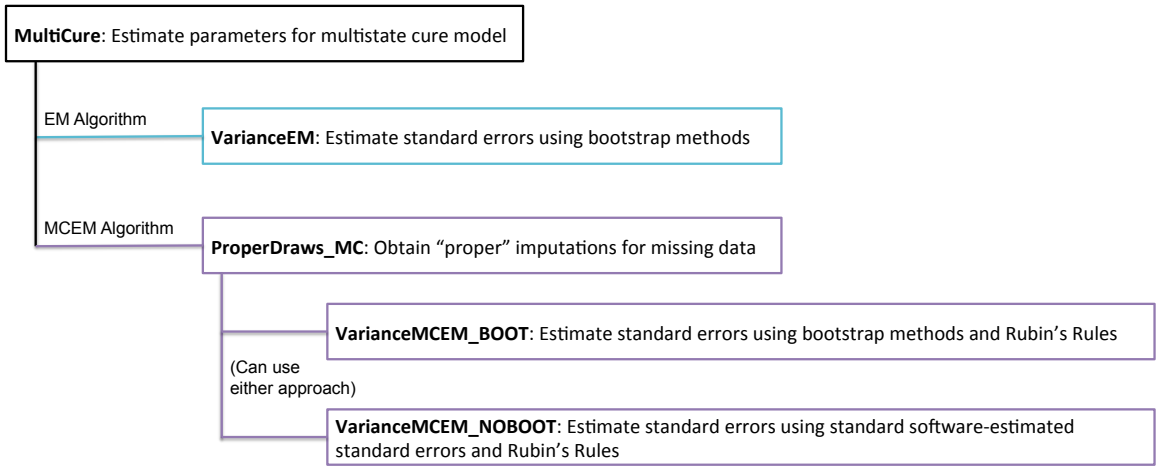
The function **MultiCure** performs the proposed EM or MCEM algorithms in order to estimate θ. The form of the baseline hazard functions is specified by the argument *BASELINE* and can take values 'weib' for Weibull baseline hazards or 'cox' for Cox baseline hazards. The restrictions on the $1 \to 4$ and $2 \to 4$ hazards can be specified with the argument *ASSUME*. Possible options include 'SameHazard' ($\Lambda_{14}(t) = \Lambda_{24}(t)$), 'SameBaseHaz' ($\Lambda^0_{14}(t) = \Lambda^0_{24}(t)$), 'ProportionalHazard' ($\Lambda_{14}(t) = e^{\beta_0}\Lambda_{24}(t)$), and 'AllSeparate' (no restrictions).

First, we need to put the data into the correct format. We create a data frame *datWIDE* containing the following variables: 'Y_R' = recurrence event/censoring time, 'delta_R' = recurrence event/censoring indicator, 'Y_D' = death event/censoring time, 'delta_D' = death event/censoring indicator, 'G' = non-cure status (NA if unknown, 1 if recurrence, 0 if assumed cured). We also create a data frame *Cov* containing the covariates we want to use in the model fit. We then create list *TransCov* to denote which covariates we want to include in which transition. *TransCov* contains elements 'Trans13', 'Trans24', 'Trans14', 'Trans34', and 'PNonCure'. Each element is a character vector containing the names of the covariates in *Cov* to be used in the model for each transition. If we wish to include no covariates in the model for a particular transition, we can include an all-zero covariate in *Cov* and list this covariate in *TransCov*. In our illustrative example, we will include both covariates in all parts of the multistate cure model.

2.1 Function Sequence

Figure 3 provides a diagram of the sequence of functions needed to obtain a complete multistate cure model fit. The parameter estimates can be obtained using the function MultiCure. The function MultiCure can estimate parameters using an EM algorithm (only missingness in cure status) or an MCEM algorithm (missingness in cure status and missingness in covariates and/or unequal censoring of the outcomes). If we fit using an EM algorithm, corresponding standard errors can be obtained using the function VarianceEM. If we fit using an MCEM algorithm, we first obtain "proper" imputations for the missing data using the function ProperDraws_MC. Using the imputed data obtained from the functionProperDraws_MC, we can then estimate standard errors using either function VarianceMCEM_BOOT or function VarianceMCEM_NOBOOT.

Figure 3: Function Sequence for Fitting Multistate Cure Model



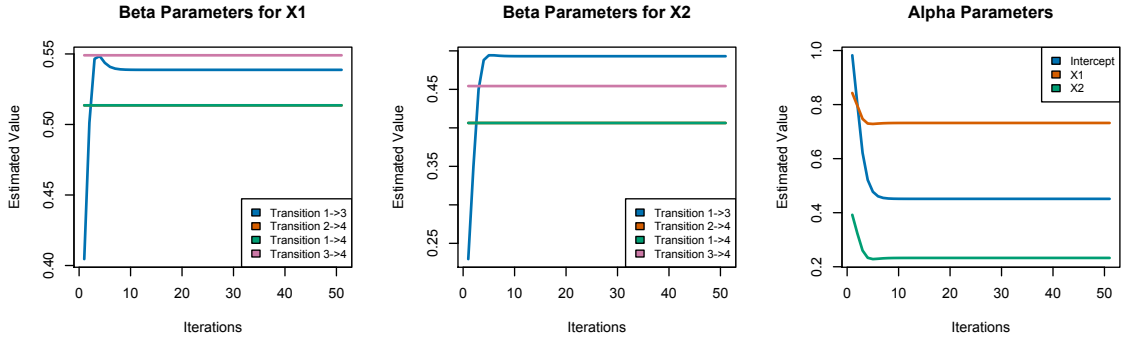
2.2 Fitting the Multistate Cure Model using an EM Algorithm

We first consider the setting with fully-observed coveriates and equal follow-up for all subjects. Suppose we want to fit a multistate cure model to the complete data assuming Weibull baseline hazards and equal $1 \to 4$ and $2 \to 4$ hazards. We can perform the proposed EM algorithm to fit the multistate cure model in R as follows:

```
Cov = data.frame(X1,X2)
VARS = names(Cov)
TransCov = list(Trans13 = VARS, Trans24 = VARS, Trans14 = VARS,
                Trans34 = VARS, PNonCure = VARS)
datWIDE = data.frame( Y_R, Y_D, delta_R , delta_D, G)
fit = MultiCure(iternum, datWIDE, Cov, ASSUME = 'SameHazard',
                TransCov = TransCov, BASELINE = 'weib')
```

Object *fit* will be a list containing the parameter estimates at the final iteration and the estimates at all previous iterations. Argument *iternum* indicates the number of iterations the algorithm will perform. We can examine the parameter estimates across iterations to evaluate the model convergence. **Figure 4** shows the trace plots for our simulated dataset.

Figure 4: Convergence of Multistate Cure Model Parameters across Iterations



We can see that the parameter estimates converge very quickly. The parameters for transition $2 \to 4$ are assumed to be equal to the parameters for transition $1 \to 4$, so the corresponding lines are the same.

Not that we have obtained an estimate for θ, we now want to estimate corresponding standard errors. The function **VarianceEM** performs variance estimation using bootstrap methods when no covariate missingness or unequal follow-up is present. We can use the following code:

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

The argument *bootnum* corresponds to the number of bootstrap samples, and *iternum* corresponds to the number of iterations of the EM algorithm used for each bootstrap sample. The object OUT is a data frame containing the variance estimates for each model parameter.

2.3 Fitting the Multistate Cure Model using an MCEM Algorithm

Suppose instead that we impose missingness in the second covariate. Then, we can fit a multistate cure model using the proposed MCEM algorithm using **MultiCure** with additional arguments. The MCEM algorithm involves imputing values for the missing covariate at each iteration of the model fitting algorithm, and the user must specify a function **COVIMPUTEFUNCTION** for performing this imputation. The user-specified function must have the following arguments: 'datWIDE', 'param', 'ImputeDat', and 'TransCov'. When the user-specified function is called within **MultiCure**, these take the following values:
(1) *datWIDE*: this is the same as in **MultiCure**
(2) *param*: a vector containing the most recent model parameters.
(3) *ImputeDat*: a list with elements [[1]], a binary vector taking value 1 if subject has unequal follow-up; [[2]], a binary matrix taking value 1 if the corresponding element of *Cov* is missing. The list also contains *single* imputations of *Cov*, G, Y_R, and δ_R from the previous iteration in slots [[3]]-[[6]] respectively. If *BASELINE* = 'cox' in **MultiCure**, this list will also contain the most recent estimate of the baseline hazards for each transition.
(4) *TransCov*: this is the same as in **MultiCure**

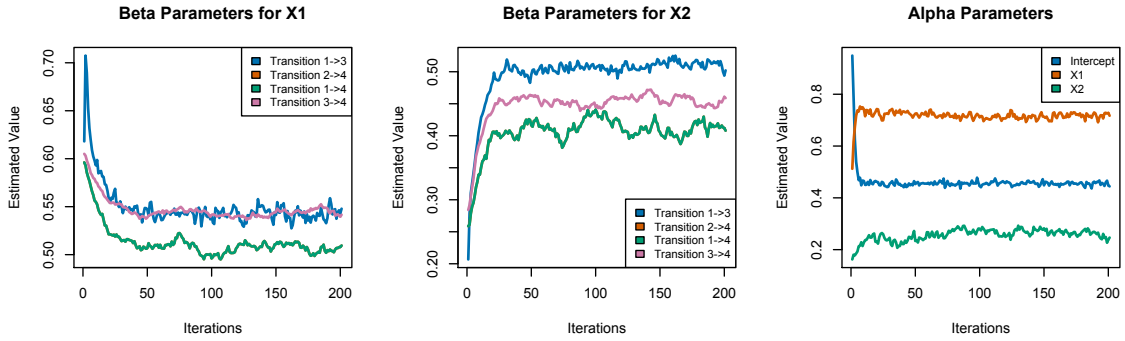
COVIMPUTEFUNCTION must return a data frame containing a *single* imputation of *Cov*. The user can use any method he/she wants to perform the imputation. In the main paper, we use a method similar to the method proposed in Bartlett *and others* (2014). Alternatively, the user could call the function *mice* (from package mice in R) or another standard imputation software within **COVIMPUTEFUNCTION**.

The user must also specify a function **COVIMPUTEINITIALIZE** for initializing the missing values of *Cov*. This has argument (S1) *Cov* (same as **MultiCure**) and (S2) *CovMissing* (binary matrix taking value 1 if corresponding element of *Cov* is missing). This function must return a single initialized version of *Cov*. We can then fit a multistate cure model using the following:

```
fit = MultiCure(iternum = 200, datWIDE, Cov, ASSUME = 'SameHazard',
        TransCov = TransCov, BASELINE = 'weib', IMPNUM = 10,
        COVIMPUTEFUNCTION = userfunc1, COVIMPUTEINITIALIZE = userfunc2)
```

The argument *IMPNUM* is an integer corresponding to the number of imputed datasets to obtain. As before, object *fit* will be a list containing the parameter estimates at the final iteration and the estimates at all previous iterations. It will also contain the imputed version of the data at the final iteration. We can again examine the parameter estimates across iterations to evaluate the model convergence. **Figure 5** shows the trace plots from the MCEM algorithm applied to our simulated dataset.

Figure 5: Convergence of Multistate Cure Model Parameters across Iterations



We can obtain an estimate of θ by taking the mean of the parameter estimates from the last few iterations of the MCEM algorithm. In order to estimate the standard errors, we first perform a post-processing step that converts the "improper" imputed data from the MCEM algorithm into "proper" imputations as defined in Little and Rubin (2002). We can use the function **ProperDraws_MC** as follows:

```
Proper = ProperDraws_MC(datWIDE,Cov, CovImp = fit[[9]], GImp = fit[[10]],
        YRImp = fit[[11]], deltaRImp = fit[[12]],
        COVIMPUTEFUNCTION = userfunc1,  COVIMPUTEINITIALIZE = userfunc2,
        ASSUME = 'SameHazard', TransCov, BASELINE = 'weib')
```

The object *Proper* is a list containing the updated imputed data. We can then estimate the standard errors overall by estimating the standard errors separately for each imputed dataset and then using Rubin's rules to obtain the overall standard errors. Within each imputed dataset, we can estimate standard errors using software to fit proportional hazards and logistic regression models or using bootstrap methods. Variance estimation using bootstrap methods can be performed using:

```
OUT = VarianceMCEM_BOOT(fit, bootnum = 50, datWIDE,
        CovImp = Proper[[1]], GImp = Proper[[2]], YRImp = Proper[[3]],
        deltaRImp = Proper[[4]],  ASSUME = 'SameHazard', TransCov,
        BASELINE = 'weib')
```

Variance estimation using standard errors from logistic/proportional hazards regression fits to the imputed data can be performed using:

```
OUT = VarianceMCEM_NOBOOT(fit, datWIDE, CovImp = Proper[[1]],
        GImp = Proper[[2]], YRImp = Proper[[3]], deltaRImp = Proper[[4]],
        ASSUME = 'SameHazard', TransCov, BASELINE = 'weib')
```

The second variance estimation method is faster than the first, but otherwise either method can be used. The object *OUT* will contain the variance estimates for $\hat{\theta}$.

Suppose we had unequal censoring instead of or in addition to covariate missingness. We

use the same code as presented above except that in the case of fully-observed covariates, the covariate imputation and initialization functions may be left unspecified.

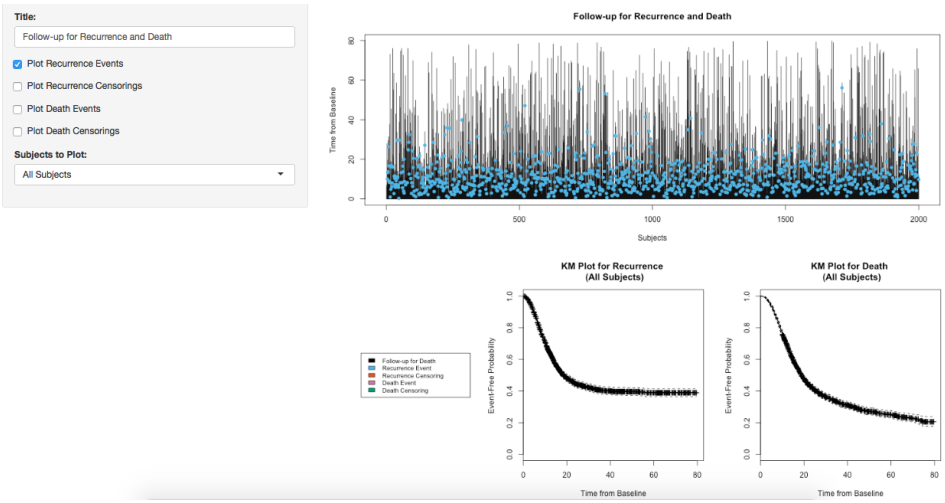
2.4 Additional Tools

In addition to functions for fitting the multistate cure model, we provide functions for data visualization prior to fitting and prediction based on the multistate cure model fit.

2.4.1 Outcome Data Visualization

In terms of visualization, we have created an RShiny application that provides the Kaplan-Meier estimator applied to time to recurrence and overall survival. Additionally, the application allows the user to create a plot showing the follow-up for overall survival for each subject. The user can then plot event and censoring times for recurrence and death, allowing the user to visualize when the events and censorings of each type are occurring. This plot can be generated using the entire analytical sample or restricting to the subset of subjects with unequal censoring. **Figure 6** shows the application output for an example simulated dataset.

Figure 6: R Shiny App Output for Data Visualization



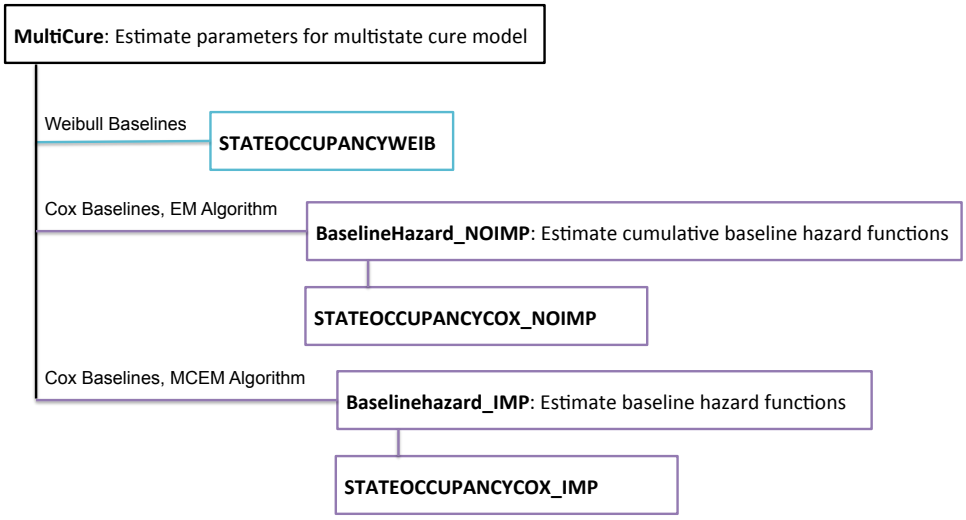
In this example, we have generated a plot comparing the observed recurrences in our analytical sample (dots) to the follow-up times for overall survival (lines). In this example data, we can see that the majority of the observed recurrences occur early in follow-up and slow down substantially by about time 40. We can run the RShiny application using the following code:

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

2.4.2 Prediction

In addition to the data visualization application, we have also developed an application for estimating the state occupancy probabilities over time given user-specified baseline covariate values based on a multistate cure model fit. We provide estimates 1) incorporating baseline covariates only and 2) incorporating baseline covariates and some post-baseline followup (except when we have Cox baseline hazards and fit using the MCEM algorithm). For a given set of baseline covariates (and potentially some post-baseline follow-up) and a particular time t, we estimate the state occupancy probability. Similar estimates for the overall survival probability and the event-free probability are also provided. The plots produced by the application can be downloaded to the working directory. Three versions of this application are provided based on the assumptions made in the multistate modeling. **Figure 7** shows the sequence of functions used to obtain the state occupancy probabilities in various modeling settings.

Figure 7: Function Sequence for Estimating State Occupancy Probabilities



Version 1: Weibull Baseline Hazards

Suppose first that the multistate cure model was fit assuming Weibull baseline hazards. We can run the RShiny app using the following code:

```
STATEOCCUPANCYWEIB(times, TransCov, newCov, beta, alpha, scale, shape)
```

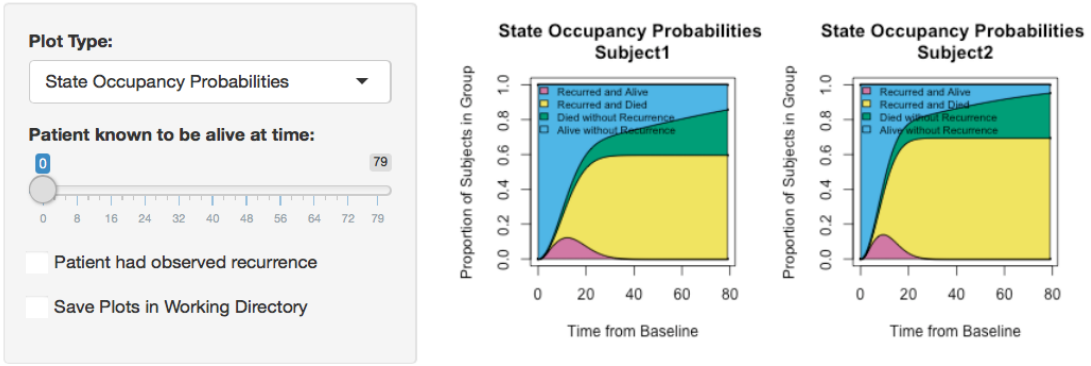
where *times* is a vector of the times at which the probabilities should be estimated, *TransCov* is the same as before, and *newCov* is a data frame containing the covariate values we want to make predictions for. The structure of *newCov* should match the structure of *Cov* from the multistate cure model fit. *beta*, *alpha*, *scale*, and *shape* are the estimated parameter values from the multistate cure model fit.

Figure 8 shows some of the application output for an example simulated dataset. **Figure 8a** shows the estimated state occupancy plots for two different subjects incorporating only

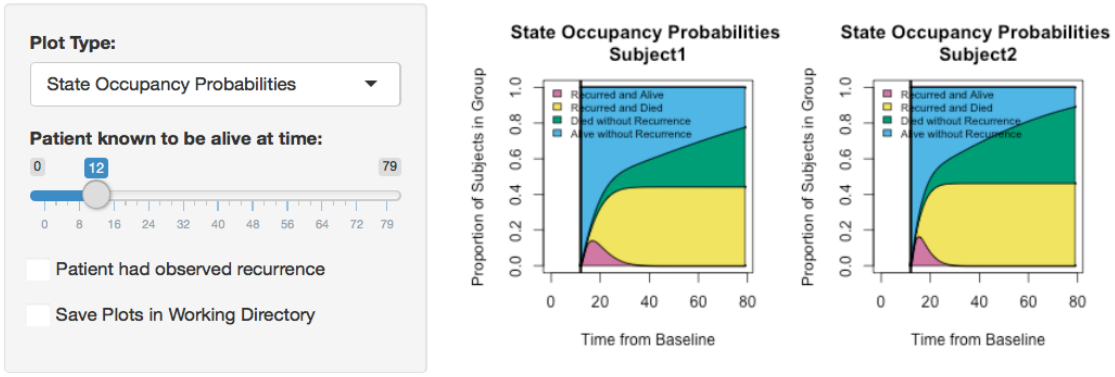
baseline information. **Figure 8b** shows the estimated probabilities assuming these same subjects were known to have been alive and recurrence-free at $t = 12$. **Figure 8c** shows the densities associated with each of the state transitions for each subject. Although not shown, the drop-down menu has additional options for displaying the overall survival probability estimates and the event-free probability estimates for each subject.

Figure 8: R Shiny App Output for Prediction

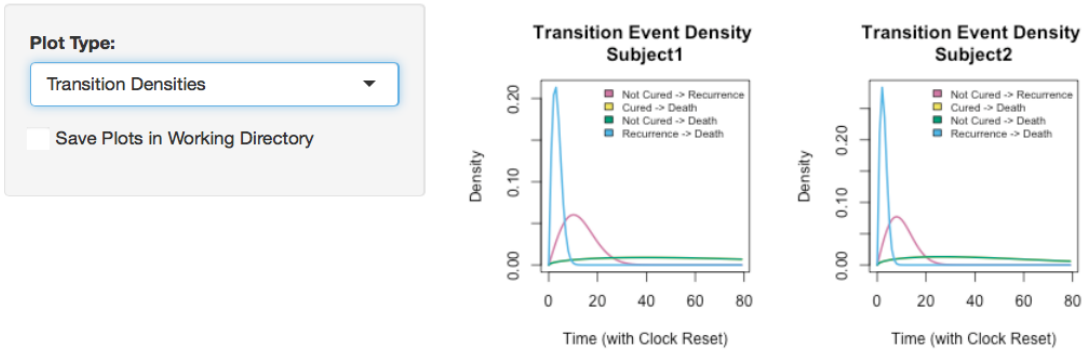
(a)



(b)



(c)



Version 2: Cox Baseline Hazards, Fit using EM Algorithm

Suppose next that the multistate cure model was fit assuming Cox baseline hazards and using data that had no missing covariates or unequal censoring. We can run the corresponding RShiny app using the following code:

```
Haz = BaselineHazard_NOIMP(datWIDE, Cov, beta, alpha, TransCov, ASSUME, p)
STATEOCCUPANCYCOX_NOIMP(times, TransCov, newCov, beta, alpha,
        Haz_13 = Haz[[1]], Haz_24 = Haz[[2]],
        Haz_14 = Haz[[3]], Haz_34 = Haz[[4]])
```

Here, the function **BaselineHazard_NOIMP** provides the estimate for the cumulative baseline hazard function for each transition at the final iteration of the EM algorithm. We use these cumulative hazards to estimate the state occupancy probabilities.*beta* and *alpha* are the estimated parameter values from the multistate cure model fit, and *TransCov*, *Cov*, and *ASSUME* are the same as in **MultiCure**. *newCov* is a data frame containing the covariate values we want to make predictions for. The structure of *newCov* should match the structure of *Cov* from the multistate cure model fit.

The output of this function is very similar to the output shown in **FigureS11** except that we provide the estimated cumulative hazard function for each transition rather than the density functions.

Version 3: Cox Baseline Hazards, Fit using MCEM Algorithm

Suppose next that the multistate cure model was fit assuming Cox baseline hazards and using data that had no missing covariates or unequal censoring. We can run the corresponding RShiny app using the following code:

```
haz = Baselinehazard_IMP(datWIDE, CovImp,GImp, YRImp,deltaRImp,
        beta, alpha, TransCov, ASSUME)
STATEOCCUPANCYCOX_IMP(times, TransCov, newCov, beta, alpha,
        Basehaz13 = haz[[1]], Basehaz24 = haz[[2]],
        Basehaz14 = haz[[3]], Basehaz34 = haz[[4]])
```

Here, the function **BaselineHazard_IMP** provides the estimate for the baseline hazard function for each transition at the final iteration of the MCEM algorithm.*beta* and *alpha* are the estimated parameter values from the multistate cure model fit, and *TransCov*, *Cov*, and *ASSUME* are the same as in **MultiCure**. *CovImp*, *GImp*, *YRImp*, and *deltaRImp* are the imputed data output by **MultiCure** from the final iteration of the MCEM algorithm. As before, *newCov* is a data frame containing the covariate values we want to make predictions for. The structure of *newCov* should match the structure of *Cov* from the multistate cure model fit. Again, the output of this function is very similar to the output shown in **Figure 8** except that we provide the estimated cumulative hazard function for each transition rather than the density functions.

References

Bartlett, Jonathan W, Seaman, Shaun R, White, Ian R and Carpenter, James R. (2014). Multiple Imputation of Covariates by Fully Conditional Specification: Accomodating the Substantive Model. *Statistical Methods in Medical Research* **24**(4), 462–487.

Conlon, A S C, Taylor, J M G and Sargent, D J. (2013). Multi-state models for colon cancer recurrence and death with a cured fraction. *Statistics in Medicine* **33**, 1750–1766.

Little, Roderick J A and Rubin, Donald B. (2002). *Statistical Analysis with Missing Data*, 2nd edition. John Wiley and Sons, Inc.