MRH: MULTI-RESOLUTION HAZARD MODELING IN R

YOLANDA HAGAR AND VANJA DUKIC

Abstract

An important aspect of survival analysis is the estimation of the hazard rate along with the effects of covariates. With recent advances in medicine, many studies now follow subjects for long periods of time, requiring flexible statistical techniques that can model long-term survival outcomes. Although there exist other packages for the modeling of the hazard rate and covariate effects, the MRH package is designed to address issues typically present in analyzing long-term survival outcomes. The multi-resolution hazard (MRH) model, as seen in [1–3, 5, 6, 9], is a Bayesian semi-parametric and computationally efficient method for flexible estimation of and inference about the hazard rate and covariate effects. The method allows for a mixture of proportional and non-proportional hazard predictors, and adapts to periods of sparsely observed failures. This vignette provides an introduction to the various features of the MRH package in R.

Key Words: MRH, hazard rate, multi-resolution hazard, non-proportional hazards, survival analysis.

1 Introduction

Advances in health care have improved the prognostic outlook of many previously incurable diseases. This is particularly evident in cancer, where once fatal illnesses are now treated as chronic conditions with long-term observable trajectories. In this instance, the long-term hazard rate estimate and its corresponding uncertainty can be critical in assessing how the risk of a disease changes over time in a population. Similarly, a variety of features may affect these long-term survival outcomes, including patient characteristics, treatment characteristics, and secular and health-care trends. These types of modern obstacles require specific software designed to accommodate long-term, and potentially large-size survival data.

The MRH is an R package designed for flexible hazard modeling, and inferrence about how patterns in disease survival change over a long-term horizon. It is able to perform:

1. Estimation and uncertainty quantification for the hazard rate (and thus the corresponding survival function and cumulative hazard).

2. Robust estimation of the hazard rate (and survival function and cumulative hazard) and the corresponding uncertainty through periods of sparsely observed failures.

3. Joint estimation of and inference about the effects of multiple predictors, under the mixture of proportional hazards (PH) and non-proportional hazards (NPH) assumptions.

The estimation and inference for the hazard rate is done via the Multi-Resolution Hazard (MRH) model, a Bayesian semi-parametric method studied in [1–3, 5, 6, 9]. The MRH model offers the ability to incorporate *a-priori* arbitrary shapes and smoothness assumptions into the hazard rate, while maintaining stable estimator properties, making it appropriate for survival analysis of long-term and possibly sparse outcomes. MRH is based on a recursive binary partition of the time axes, with a tree structure geared to support multi-timescale inference.

The MRH package in R provides the user with the estimate and the corresponding uncertainty of the baseline hazard rate, as well as the covariate effects under a mixture of PH and NPH settings. In addition, a folder of output is created with graphical diagnostics for MCMC convergence and text files of the MCMC parameter chains. There are additional functions available to the user including those for model comparison, various plotting features, and examining the model fit.

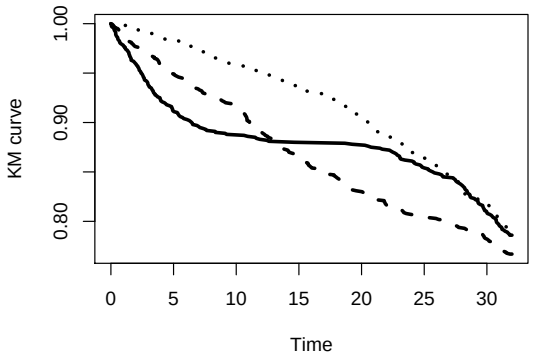

Figure 1: The Kaplan-Meier curve of the survival rates for the three different treatment groups in the simulated ‘‘MRHdata’’ example. The graph indicates that the covariate should not be modeled under the proportional hazards assumption. Estimates for the Kaplan-Meier curves were made using the ‘‘survfit" function in the survival package in R.

2 Example

For the purpose of illustrating all of the features of the MRH package, we have simulated a dataset designed to resemble a long-term cancer study. This data set has high rates of censoring and periods with sparsely observed failures. In addition, one of the covariates (treatment) does not follow a proportional hazards assumption. This dataset is named MRHdata and has been included as a part of the MRH package.

Each patient's failure time is affected by a treatment variable (a nominal variable with 3 levels, named treatment in the MRHdata dataset), as well as age (a continuous and standardized variable, named age in the MRHdata dataset), and gender (a binary variable, named gender in the MRHdata set). There are 1000 subjects in each treatment group ($n = 3000$ total), and the treatment-specific hazard functions are non-proportional. The Kaplan-Meier curve of the survival rates for these three different treatment groups is shown in Figure 1, which was created using the survfit function in the survival package.

The simulated data set can be loaded into R as follows:

```
# Read in the data set:
data(MRHdata)
MRHdata[1:5,]
       time delta gender treatment        age
1  2.665168     1      0         0 -0.8938726
2 32.000000     0      0         0  0.8390768
3 32.000000     0      0         0 -1.8786602
4  1.614175     1      0         0 -2.3418080
5 32.000000     0      0         0  0.6862053
```

In the MRHdata dataset, the time variable is the time from the inception of treatment to failure, and delta denotes the censoring variable, which equals '1' if the failure event is observed, and '0' if the failure time is censored. The gender variable is binary, with '0' for males and '1' for females.

As would be typical with long-term follow-up, the observed failure rates towards the end of the study period are low, and failure events are sparse. The percentage of subjects with observed failures after the study mid-point is about 11% (combined across all three treatment groups), and after the 3/4 study point is about 6.5%. The overall censoring rate is almost 80%. Details of the data set can be found in Table 1.

3 Fitting the MRH Model

Once the data set is loaded into R, there are two stages to fitting the MRH model. The first is to select the desired "time-resolution", i.e., the number of time intervals or "bins". The second is to fit the model with the desired subset and form of predictors.

		All subjects		Treatment Type						Censored	
				0		1		2			
		N	%	N	%	N	%	N	%	N	%
Total number of subjects		3000	33.33	1000	33.33	1000	33.33	1000	33.33	2344	78.13
Number censored		2344	78.13	786	26.20	767	25.57	791	26.37		
Gender	Male	1500	50.00	500	16.67	500	16.67	500	16.67	1116	37.20
	Female	1500	50.00	500	16.67	500	16.67	500	16.67	1228	40.93
		Mean	SD	Mean	SD	Mean	SD	Mean	SD	Mean	SD
Age		0.00	1.00	-0.02	1.00	0.01	0.99	0.01	1.01	-0.06	0.99
Time to failure		28.45	8.30	28.17	9.24	27.8	8.82	29.38	6.52	31.98	0.67

Table 1: Characteristics of the 3000 patients in the simulated `MRHdata` dataset. Percentages shown are out of the total number (3000) of subjects. Age and time to failure, which are continuous variables, are reported in terms of the mean and standard deviation.

3.1 Selecting the time resolution (specifying M)

The MRH methodology is based on a binary partition, meaning that it divides the total study time into 2^M time intervals, or bins. While the bins need not be equal-sized, equal length is recommended for convenience in the absence of any *a-priori* desired partition. The choice of M will be most often determined through biological rationale, and based on the *clinically or biologically* desired time resolution. For example, in cancer studies, the time resolution might be chosen to match the desired follow-up visit schedule. In other cases, the choice of M can be based on penalized likelihood criteria (such as AIC, BIC), letting the data guide the optimal bin size. The choice of bins is discussed in more detail in [1, 2, 6].

In some cases the user might wish to explore different bin lengths. In these instances, the `FindBinWidth()` function can help. Example code, using the simulated `MRHdata`, is given below:

```
##############################################################################
# CODE TO EXAMINE THE BIN LENGTH WITH DIFFERENT M VALUES
##############################################################################
#
# Find the bin widths for values of M ranging from 2 to 10.  Specify the time unit as weeks
# by setting "time.unit" equal to 'w'.  Other options are seconds (s), minutes (min), hours (h),
# days (d), weeks (w), months (mon), and years (y).

data(MRHdata)
FindBinWidth(MRHdata$time, MRHdata$delta, 'w')
[1] The mean failure time is  29.839 weeks
[1] The median failure time is  33.252 weeks
[1] The range of failure times is  0.026 to 35.649 weeks

          secs       mins       hours       days      weeks      months         years
M2  5390102.43 89835.0405 1497.250675 62.3854448 8.91220640 2.049624469  0.1708020391
M3  2695051.21 44917.5202  748.625337 31.1927224 4.45610320 1.024812235  0.0854010196
M4  1347525.61 22458.7601  374.312669 15.5963612 2.22805160 0.512406117  0.0427005098
M5   673762.80 11229.3801  187.156334  7.7981806 1.11402580 0.256203059  0.0213502549
M6   336881.40  5614.6900   93.578167  3.8990903 0.55701290 0.128101529  0.0106751274
M7   168440.70  2807.3450   46.789084  1.9495451 0.27850645 0.064050765  0.0053375637
M8    84220.35  1403.6725   23.394542  0.9747726 0.13925322 0.032025382  0.0026687819
M9    42110.18   701.8363   11.697271  0.4873863 0.06962661 0.016012691  0.0013343909
M10   21055.09   350.9181    5.848635  0.2436931 0.03481331 0.008006346  0.0006671955
```

The results show the bin length for each value of M for the different units of time. For example, with $M = 3$, the bin length will be approximately 1 month, which could be useful in interpretation of results. Alternatively, the range of the failure times goes to 35.6 weeks; this is close to 32 ($= 2^5$) weeks, implying that $M = 5$ could be appropriate. In that case, the failure times past 32 weeks would be considered censored. If needed, the same routine can be used again, setting the "maxStudyTime" option equal to 32. The routine will re-calculate the bin widths, and also

calculate the number of additional observed failures that would be censored if the maximum study time were lowered to 32 weeks.

```
# Re-examine the bin widths with a maximum study time equal to 32
FindBinWidth(MRHdata$time, MRHdata$delta, 'w', maxStudyTime = 32)
[1] With the given study length,  0 extra observed failures will be censored.

[1] The mean failure time is  28.45 weeks
[1] The median failure time is  32 weeks
[1] The range of failure times is  0.026 to 32 weeks

       secs  mins   hours     days   weeks      months        years
M2  4838400 80640 1344.00 56.00000 8.00000 1.839835729 0.1533196441
M3  2419200 40320  672.00 28.00000 4.00000 0.919917864 0.0766598220
M4  1209600 20160  336.00 14.00000 2.00000 0.459958932 0.0383299110
M5   604800 10080  168.00  7.00000 1.00000 0.229979466 0.0191649555
M6   302400  5040   84.00  3.50000 0.50000 0.114989733 0.0095824778
M7   151200  2520   42.00  1.75000 0.25000 0.057494867 0.0047912389
M8    75600  1260   21.00  0.87500 0.12500 0.028747433 0.0023956194
M9    37800   630   10.50  0.43750 0.06250 0.014373717 0.0011978097
M10   18900   315    5.25  0.21875 0.03125 0.007186858 0.0005989049
```

Thus, if the maximum study time is lowered to 32 weeks, we can set $M = 5$, with each bin representing one week. In this example, there are zero additional observed failures that will be censored, so the information loss will be minimal. We choose to adjust our data for further use:

```
# Adjust the data so that the maximum study time is equal to 32
MRHdata$delta[MRHdata$time > 32] = 0
MRHdata$time[MRHdata$time > 32] = 32
```

While the above commands result in little change for the current dataset, such study duration adjustment may have important consequences in other datasets. Similarly, the maximum study time can also be increased to values greater than the last observed study time. However, this is rarely recommended, as there will be no observed information in the added period of time.

3.2 Fitting the MRH models

Once M has been determined, the MRH model is fit. In the examples below, we have set M equal to 5, with each bin now representing one week. We will examine two types of models: both will include treatment, age, and gender as covariates. However, in the first model, treatment will be included under the proportional hazards assumption, and in the second model, treatment will be included under the non-proportional hazards assumption.

3.2.1 Proportional hazards assumption

While the `MRHdata` has been simulated with non-proportional treatment hazards, for expository purposes we first show how to fit an MRH model with the proportional hazards assumption for all covariates. The formula for the model follows the same pattern as the standard right-censored survival model with age, gender, and treatment covariates:

```
formula = Surv(time, delta) ~ age + gender + treatment
```

The model is fit using the `estimateMRH()` function as follows:

```
###### CODE FOR FITTING THE MRH PROPORTIONAL HAZARDS MODEL ######
data(MRHdata)
fit.PH = estimateMRH(Surv(time, delta) ~ age + gender + as.factor(treatment), data = MRHdata,
  M = 5, outfolder = 'MRHresultsPH', maxIter = 5000)
```

Immediately upon typing the above command, the routine estimates the total run time. The estimated total run time is an approximation based on the first 100 iterations of Gibbs sampler:

```
[1] "MCMC routine running.  Calculating estimated runtime for 5000 iterations..."
[1] "Estimated total run time is 10 minutes"
```

The user will also be notified every 5,000 iterations the routine completes.

The `estimateMRH()` routine will run until the maximum number of iterations is reached (specified through the "maxIter" option), or until the chains show evidence of convergence, whichever occurs first. The routine automatically checks for evidence of convergence every 100,000 iterations. If the maximum number of iterations to a value smaller than 100,000, convergence diagnostics are performed upon the completion of the run.

The `estimateMRH()` routine can also automatically thin the chains, and delete the burn-in period of the chains. The user specifications on these values can be made through the "thin" option (default value is 10), and the "burnIn" option (default is the minimum of 50,000 or half of "maxIter"). Details on setting and/or fixing the chain specifications and convergence checking can be found in Section 4.3.

In the above example, all model output is placed into a newly created sub-folder (within the current working directory), named "MRHresultsPH". This subdirectory name is specified by the "outfolder" option in the `estimateMRH()` command. However, if that name is omitted, the default output folder name will be "MRHresults". Please note that the folder name can be added to a pathname, but the path must be accessible from the working directory.

3.2.2 Non-proportional hazards assumption

As seen in Figure 1, the Kaplan-Meier curves indicate that the treatment effect might be non-proportional. For that reason, we now modify the above model, specifying the treatment variable as a non-proportional effect in the `estimateMRH()`, using the `nph` function in the formula statement:

```
formula = Surv(time, delta) ~ age + gender + nph(treatment)
```

such that the model is fit as follows:

```
###### CODE FOR FITTING THE MRH NON-PROPORTIONAL HAZARDS MODEL ######
fit.NPH = estimateMRH(Surv(time, delta) ~ age + gender + nph(treatment), data = MRHdata,
  M = 5, outfolder = 'MRHresults_NPH', maxIter = 5000)
[1] "MCMC routine running.  Calculating estimated runtime for 5000 iterations..."
[1] "Estimated total run time is 15 minutes"
```

An important item to note is that the non-proportional covariate must be a nominal categorical variable. More than one non-proportional covariate can be entered in to the model, using repeated `nph()` functions in the formula. In the case of multiple non-proportional predictors, the routine will merge the NPH variables into a single interaction variable and separate hazards will be estimated for each combination of levels. For example, if both `gender` (2 categories) and `age` (separated into three categories) are entered as non-proportional covariates, the `estimateMRH()` routine will create a new `age*gender` variable with 6 categories, pairing each `gender` with each `age` bracket. Consequently, six separate hazard rates will be estimated, one for each `age` and `gender` combination. Interaction categories containing zero subjects will be omitted. The user may also choose to enter into the `nph` function a single variable, with levels pre-defined by the user, which partitions the population into the separate hazard groupings.

Convergence checking, output generation, and other specifications on the MCMC chains are the same as with the PH model.

3.3 Pruning the MRH tree

During periods when few failures are observed, such as at the end stages of a clinical trial with a long follow-up period, the estimator of the hazard rate may not be efficient. Because of this, the user has an option to "prune" the MRH tree, and merge the time intervals (bins) containing sparsely observed failures. Pruning can increase the efficiency and decrease the parameter vector dimension *a priori*, thus additionally decreasing the length of time required for the MCMC routine to run. This can be done manually by entering a user-defined pruning vector, or

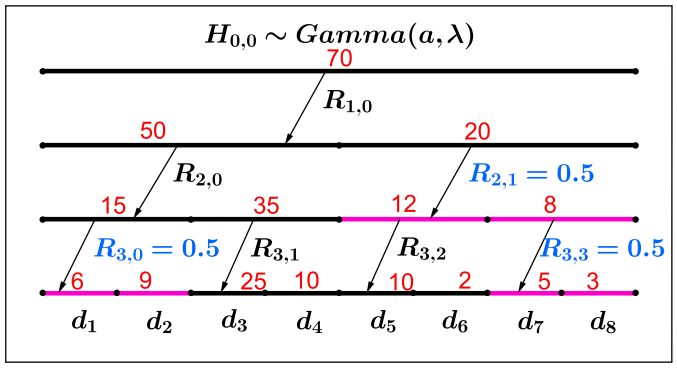

Figure 2: A diagram of the MRH prior in the case of $M = 3$ (i.e. 8 bins). $H_{0,0}$, which has a $Gamma(a, \lambda)$ prior, denotes the cumulative hazard at the end of the study period. The $R_{m,p}$ denote the "splits" for the MRH tree, with 0.5 representing *a-priori* equal hazard rates between two adjacent bins within the m^{th} level of the tree. Similarly, $R_{m,p} > 0.5$ denotes a higher hazard rate in the first of the adjacent bins (with a lower hazard rate in the second bin), and $R_{m,p} < 0.5$ denotes a lower hazard rate in the first of the adjacent bins (with a higher hazard rate in the second bin). The $d_j, j = 1, \ldots, J$ denote the hazard increments (i.e. the cumulative hazard rate within each bin) for bins 1 through 8. The sections colored in pink are those that can be merged through the pruning algorithm, so the corresponding $R_{m,p}$ values are set equal to 0.5. In the models presented in this vignette, M is equal to 5, and we examine only pruning the bottom two levels, and pruning all the levels of the multiresolution prior.

through the use of our data-driven pruning function that detects and combines intervals where the failure rates are statistically similar (see [3]) . This tool performs data-driven merging of time intervals with similar hazard rates, resulting in increased computational efficiency and reduced uncertainty in hazard rate estimation over regions with low event counts. Details on the pruning algorithm can be found in [3]. This vignette only covers the use of the pruning algorithm; however, details on the creation and use of pruning vectors can be found in the help pages of the `estimateMRH()` and `Prune()` functions.

We will cover two types of pruned models: one with all MRH tree levels subject to pruning, and one with only the bottom two levels subject to pruning (see Figure 2). Code for running the pruning function on the PH model is shown below.

```
##############################################################################
# CODE FOR FITTING THE PRUNED MODELS
##############################################################################

# Fit the MRH PH model, with all levels subject to pruning
#(only requires the "prune" option to be set equal to TRUE)

fit.PH.pruneall = estimateMRH(Surv(time, delta) ~ age + gender + as.factor(treatment),
 data = MRHdata, M = 5, outfolder = 'MRHresults_PH_pruneall', maxIter = 5000,
 prune = TRUE)


# Fit the MRH PH model, pruning only the bottom two levels
# (set "prune" equal to TRUE, and set "prune.levels" equal to 2)

fit.PH.prune2 = estimateMRH(Surv(time, delta) ~ age + gender + as.factor(treatment),
 data = MRHdata, M = 5, outfolder = 'MRHresults_PH_prune2', maxIter = 5000,
 prune = TRUE, prune.levels = 2)
```

Similar code can be used to prune the NPH model: the pruning will be performed separately for the hazard rate of each treatment group. Convergence checking, output generation, and other specifications on the MCMC chains are the same as with the non-pruned models, and details are discussed in the sections that follow.

Figure 3 shows the estimated hazard rate of the pruned models using `MRHdata`, contrasted against the original PH model created in the previous section. While the unpruned model allows many sharp peaks in the hazard rate, its credible intervals in each bin are large. In the pruned models, the U-shape of the baseline hazard rate is still retained, although the sharp changes are smoothed over. In addition, the 95% credible intervals are narrower. In this

instance, differences between the two-level pruning and all-level pruning appear to be minimal. The computational run time for the fully-pruned model was approximately eight times shorter than for the unpruned model.

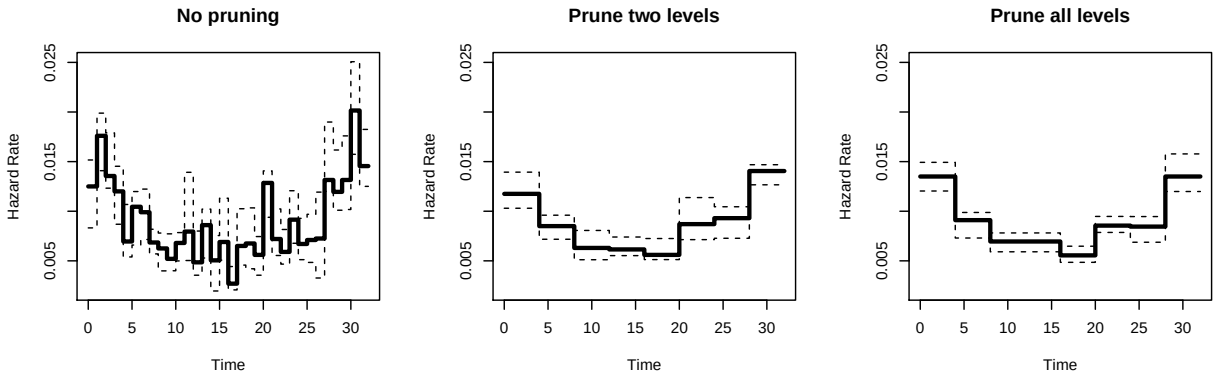

Figure 3: Comparison of the estimated hazard rate for three pruned models: no pruning (left), the last two levels pruned (center), and all levels pruned (right). The pruned models both have smaller credible intervals, and smooth over some of the sharp changes in the hazard rate while still capturing the overall shape.

Pruning the MRH prior is a compromise between the desire for detail and the amount of data; as the resolution increases (and the number of time bins increases), counts within each bin will decrease. While useful for revealing detailed patterns, a large number of bins (and consequently, a large number of model parameters) will generally require longer computing times and will cause the event counts in each bin to decrease, translating into lower efficiency. The user should always carefully consider this trade-off in making the decision to prune the MRH tree.

3.4 MRH tree parameters k and γ

The prior for the $R_{m,p}$ (i.e. the "split") parameters is a beta distribution, with the form:

$$R_{m,p} \sim Beta\left(2\gamma_{m,p}k^m a, 2(1-\gamma_{m,p})k^m a\right).$$

Below is a brief explanation of the assumptions that stem from using different fixed or sampled values of k and γ. As specific values of k and/or γ are linked with different *a-priori* shapes and smoothness of the hazard rate, using values other than 0.5 (the default values) or sampling the parameters is also helpful for purposes of sensitivity analyses. More details on the role of these hyperparameters can be found in [1–3].

3.4.1 Comparison and selection of k

The default value for k is 0.5, which implies zero *a-priori* correlation among the hazard increments within each bin. When $k > 0.5$, the increments are positively correlated *a priori*. Similarly, when $k < 0.5$, the hazard increments are negatively correlated *a priori*. Another way to consider k is that higher values lead to "smoother" functions, as the split parameters are more concentrated around 0.5 resulting in a more "even" distribution of hazard across time.

The user may change the default value of $k = 0.5$ by entering a value for"k.fixed," with $k \in (0, \infty)$, in the `estimateMRH` function. Note that in the NPH models, a vector of k values can be entered, with one k for each subgroup hazard rate. However, if only one value of k is specified, that value will be used for all hazard rates. Alternatively, if "k.fixed" is set to `FALSE`, the routine will sample the k parameter(s), putting an exponential hyper prior on k.

Below is example code for comparing four pruned models with different methods for comparing k: 1) fix $k = 0.5$ (default), 2) fix $k = 0.2$, 3) fix $k = 1.0$, and 4) sample k (i.e. estimate k based on the observed data). The code examines how the fixed k values effect the estimates with the full data set, and then on a smaller subset of the data with 300 subjects. These results are matched to the distribution of k when it is sampled. This comparison shows that

while the fixed hyperparameter values do not play a large role when the sample size is large, they can dramatically change the estimates on smaller sized data sets (see Figure 4), especially in the presence of heavy censoring.

```
fit.keq.2 = estimateMRH(Surv(time, delta) ~ age + gender + as.factor(treatment),
data = MRHdata, M = 5, outfolder = 'MRHresults_keqpoint2', maxIter = 5000,
prune = TRUE, prune.levels = 2, k.fixed = 0.2)

fit.keq1 = estimateMRH(Surv(time, delta) ~ age + gender + as.factor(treatment),
data = MRHdata, M = 5, outfolder = 'MRHresults_keq1', maxIter = 5000,
prune = TRUE, prune.levels = 2, k.fixed = 1)

fit.ksample = estimateMRH(Surv(time, delta) ~ age + gender + as.factor(treatment),
data = MRHdata, M = 5, outfolder = 'MRHresults_ksample', maxIter = 1000,
prune = TRUE, prune.levels = 2, k.fixed = FALSE)

#### Same as above, but on a small subset of 300 subjects
subset = c(sample(1:500, size = 50), sample(1:500+500, size = 50),
sample(1:500+1000, size = 50), sample(1:500+1500, size = 50),
sample(1:500+2000, size = 50), sample(1:500+2500, size = 50))

# Set fixed value of k equal to 0.2
fit.keq.2_subset = estimateMRH(Surv(time, delta) ~ age + gender + as.factor(treatment),
data = MRHdata[subset,], M = 5, outfolder = 'MRHresults_keqpoint2_subset', maxIter = 5000,
prune = TRUE, prune.levels = 2, k.fixed = 0.2)

fit.keq.5_subset = estimateMRH(Surv(time, delta) ~ age + gender + as.factor(treatment),
data = MRHdata[subset,], M = 5, outfolder = 'MRHresults_keqpoint5_subset', maxIter = 5000,
prune = TRUE, prune.levels = 2)

fit.keq1_subset = estimateMRH(Surv(time, delta) ~ age + gender + as.factor(treatment),
data = MRHdata[subset,], M = 5, outfolder = 'MRHresults_keq1_subset', maxIter = 5000,
prune = TRUE, prune.levels = 2, k.fixed = 1)

fit.ksample_subset = estimateMRH(Surv(time, delta) ~ age + gender + as.factor(treatment),
data = MRHdata[subset,], M = 5, outfolder = 'MRHresults_ksample_subset', maxIter = 1000,
prune = TRUE, prune.levels = 2, k.fixed = FALSE)
```

3.4.2 Comparison and selection of $\gamma_{m,p}$

The mean of the prior for each split parameter $R_{m,p}$ is $E(R_{m,p}) = \gamma_{m,p}$, with $\gamma_{m,p} \in (0, 1)$. This allows the user to *a priori* "center" the baseline hazard increments in each bin at a desired value (please see [1] for details).

The default value for the γ parameters is 0.5, however the user may change the default values via the "gamma.fixed" option. In general, unless the user has specific information that would help in the adjustment of the γ parameters, we recommend keeping the value fixed at 0.5. In the case of user input, a $\gamma_{m,p}$ value must be specified for each of the split parameters. In NPH models, a matrix of fixed γ values may be entered, with each column representing the desired values for each subgroup hazard rate. However, this is not required; if only a vector is specified, that vector will be used for all hazard rates. A user may also decide to sample the γ parameters by setting "gamma.fixed" to `FALSE`, putting a beta hyper prior on each $\gamma_{m,p}$.

Code for fitting the model by adjusting or sampling γ is below.

```
# Set fixed value of gamma equal to 0.75 in PH model. Gamma is a vector with 2^M-1 bins.
fit.PH.gamma.75 = estimateMRH(Surv(time, delta) ~ age + gender + as.factor(treatment),
data = MRHdata, M = 5, outfolder = 'MRHresults_PH_gamma.75', maxIter = 1000,
gamma.fixed = rep(.75, 2^5-1))
```

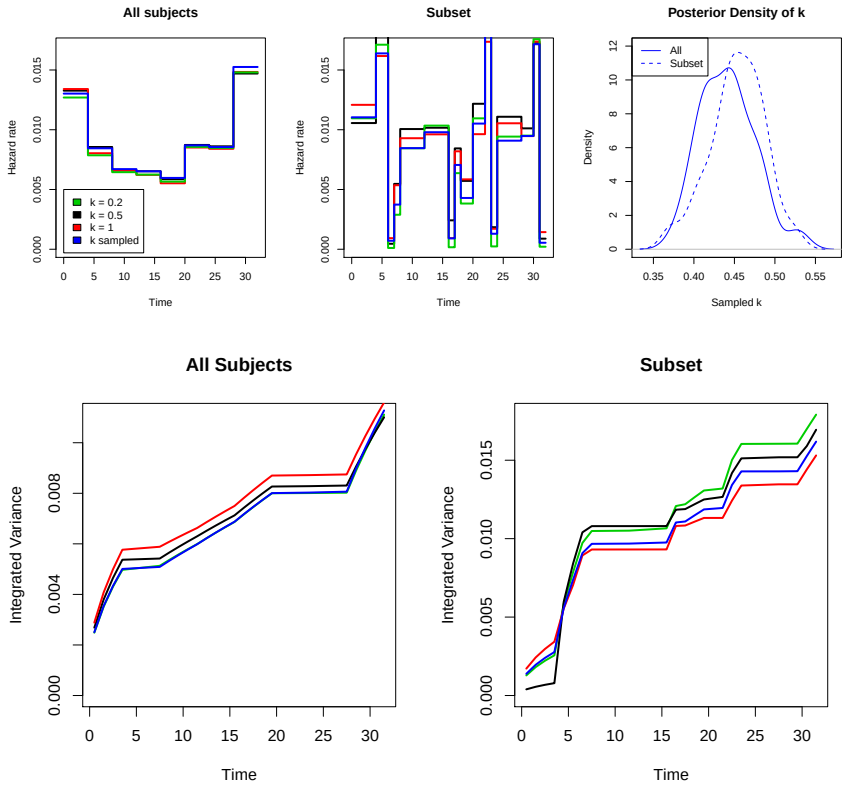

Figure 4: TOP ROW: Comparison of the effect of different k values on the hazard rate estimates, with the estimate for k from the sampled model calculated as the median of the posterior distribution for k. The graph on the upper left, which shows estimates calculated using the entire data set, shows little to no effect of the value of k, with slightly smoother estimates for larger k values. However, the graph in the center with estimates based on a subset of n = 300 subjects, shows larger differences based on the value of k. Recall that the censoring rate in this dataset is about 80%. The right-most graph shows the posterior distribution for k, obtained from the sampled models. The distributions are similar, and center around $k = 0.45$.
BOTTOM ROW: The integrated variability of the estimated hazard rate at each time point is estimated (for time interval j, using specific values of k) as $\sqrt{\sum_{\ell=1}^{j} \left(h_k(t_\ell) - \bar{h}_k\right)^2}$, where $\bar{h}_k$ is the average hazard rate estimate across all times bins for the $k = k$ estimate. Both bottom plots show that the estimated hazard rate is less varying (i.e. is "smoother") for larger values of k, with greater differences between the estimators observed when there are fewer subjects. Note that it is difficult to discern differences between the $k = 0.2$ and sampled k fitted models (left graph).

```
# Set fixed value of gamma equal to 0.75 in NPH model.
# Gamma can either be entered as a vector with 2^M-1 bins, or a matrix with 2^M-1
# rows and 3 (the number of strata in the NPH treatment variable)
fit.NPH.gamma.75 = estimateMRH(Surv(time, delta) ~ age + gender + nph(treatment),
data = MRHdata, M = 4, outfolder = 'MRHresults_NPH_gamma.75', maxIter = 1000,
gamma.fixed = rep(.75, 2^4-1))
# This will do the same thing as the code above.  It's only necessary if the
# different treatment groups need different values for fixed gammas.
fit.NPH.gamma.75 = estimateMRH(Surv(time, delta) ~ age + gender + nph(treatment),
data = MRHdata, M = 4, outfolder = 'MRHresults_NPH_gamma.75', maxIter = 1000,
gamma.fixed = matrix(.75, nrow = 2^4-1, ncol = 3))

# Set fixed values in first half of the study to 0.75,
# and in the second half of the study to 0.5
```

```
fit.NPH.gamma.mixed = estimateMRH(Surv(time, delta) ~ age + gender + nph(treatment),
data = MRHdata, M = 4, outfolder = 'MRHresults_NPH_gamma.mixed', maxIter = 1000,
gamma.fixed = c(rep(0.75, 8), rep(0.5, 7)))

# Sample the gamma values (estimate based on the observed data) for the PH model
fit.PH.gamma.sampled = estimateMRH(Surv(time, delta) ~ age + gender + as.factor(treatment),
data = MRHdata, M = 4, outfolder = 'MRHresults_PH_gamma.sampled', maxIter = 1000,
gamma.fixed = FALSE)
```

4 Model Results

The fitted MRH model object contains summaries and other useful information about the model. In addition, an output folder contains convergence diagnostics, plots of the hazard rate, and a text file containing the MCMC chains. Below are the details.

4.1 Fitted object summaries

The fitted MRH model object contains estimates and 95% credible intervals of the hazard rate, the cumulative hazard and survival function, the covariate effects, the hazard increments, and the split parameters. Estimates are reported as medians of the marginal posterior distributions. The 95% credible intervals are the central credible intervals – i.e., the 2.5th and 97.5th percentiles of the marginal posterior distributions. The posterior medians, 2.5th and 97.5th percentiles are stored as "q.5", "q.025", and "q.975" in the summary table, respectively. The `estimateMRH` routine always reports the 95% credible interval; however, the user can change and examine different α-level interval using the "summary" function (see Section 5.2 for details).

```
# Information available in the MRH fitted object
names(fit.PH)
 [1] "summary"           "d"                 "beta"              "H"
 [5] "Rmp"               "hazardRate"        "AIC"               "BIC"
 [9] "DIC"               "burnIn"            "thin"              "TotalIters"
[13] "convergence"       "gelman.rubin.used" "fix.thin"          "fix.burnIn"
[17] "fix.max"           "initialValues"     "runtime"           "outfolder"
[21] "maxStudyTime"


fit.PH$hazardRate
            q.5     q.025     q.975
h.Bin1  0.01150 0.0086500 0.0137750
h.Bin2  0.01345 0.0105875 0.0194500
. . .
h.Bin32 0.01280 0.0084875 0.0158250
```

In addition to the parameter estimates, the fitted MRH object also contains AIC, BIC, and DIC values (useful for model comparison), the final burn-in and thinning values, the total number of iterations, the initial parameter values, and other useful information about the model. The total number of iterations is the total number iterations performed by the MCMC routine, and includes all values that are thinned and burned. Details can be found in the help pages for `estimateMRH`.

4.2 Model output folder

In addition to the summaries contained in the fitted MRH model object (eg. `fit.PH` from the PH example), multiple output files are created and placed in a result subfolder (newly created within the current working directory) labeled "MRHresults" by default (the name can be changed using the "outfolder" option, such as "MRHresultsPH" in the PH example). When the `estimateMRH` routine has finished running, the output folder holds three items:

1. A file titled titled "MCMCchains.txt" containing the thinned MCMC chains with the burn-in period removed, the MCMC iteration number, and the log-likelihood value at each iteration. The text file can be read into R in separate sessions, making it useful in situations where the MRH routine is run as a background job (for details on using these text files, see section 6).

2. Diagnostic MCMC convergence graphs (see Section 4.3)

3. Plots of the hazard rate(s), as shown in Figure 5.

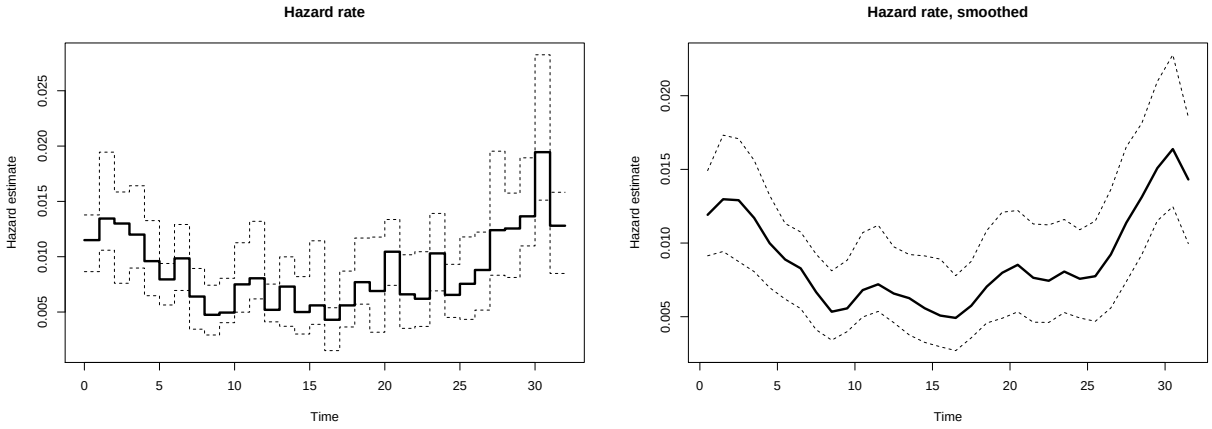

Figure 5: The estimated baseline hazard rate and 95% credible intervals for the MRH PH model. The graph on the left shows the raw estimates of the hazard rate and 95% uncertainty bounds, labeled as file "HazardRateGraph.pdf" in the output folder. The graph on the right shows the smoothed estimates of the hazard rate and 95% uncertainty bounds, labeled as file "HazardRateSmoothGraph.pdf" in the output folder. For NPH models, where more than one hazard rate is estimated, multiple hazard rate graphs are provided.

4.3 Convergence diagnostics

The `estimateMRH()` routine estimates the autocorrelation and checks for evidence of convergence in the MCMC chains every 100,000 iterations. The autocorrelation is estimated over lags equal to the current thinning value (the default starting thinning value is 10 in the `estimateMRH()` routine), as well as lags equal to 5, 10, and 15 times the current thinning value. If autocorrelation is detected, the current thinning value is increased to the next smallest possible thinning value that removes the autocorrelation. In addition, convergence is checked through the Geweke diagnostic, increasing the burn-in number if necessary. If no evidence to the contrary to convergence, then the routine will stop running and return the results. Alternatively, the routine will run for another 100,000 iterations and then analyze the chains again. This process continues until the maximum number of iterations (specified by "maxIter") is reached. The default value of the maximum number of iterations is 1,000,000. If the user specifies a maximum number of iterations less than 100,000, the diagnostics are checked when that number is reached. If the MCMC chains do not appear to have converged when the routine is finished, the user will see a warning message as follows:

```
fit.NPH = estimateMRH(Surv(time, delta) ~ age + gender + nph(treatment), data = MRHdata,
  M = 4, outfolder = 'MRHresults_NPH', maxIter = 1000)
[1] "MCMC routine running.  Calculating estimated runtime for 1000 iterations..."
[1] "Estimated total run time is 15 minutes"
[1] Estimation routine finished, calculating results....

Warning message:
In estimateMRH(Surv(time, delta) ~ age + gender + nph(treatment),  :
  Algorithm did not converge, parameter estimates may not be reliable.
  Results based on  1000 MCMC iterations.
```

The user can prevent the `estimateMRH` routine from estimating the burn-in and the thinning parameter, as well as from stopping early if the convergence has been detected before the maximum number of iterations has been run. To do this, the user would set "fix.burnIn", "fix.thin", and/or "fix.max" equal to `TRUE`, respectively. The routine will then not adjust these values, regardless of the chain convergence diagnostics. More details on how autocorrelation, the burn-in number, and convergence are determined can be found in the `estimateMRH` help file.

In addition to providing the MCMC chains in an output folder, the `estimateMRH` routine also creates figures of graphical diagnostics for convergence for each parameter. These graphics include a trace plot, smooth density plot, moving average plot, and autocorrelation plot for each parameter in the model, all saved as pdf files under the title "convergenceGraphs." If the number of model parameters is large, there may be multiple figures related to the convergence of the MCMC chains. An example of these graphics is given for the NPH example ("fit.NPH" from subsection 3.2.2) in Figure 6, which shows "convergenceGraphs26.pdf" from the "MRHresults_NPH" output folder. We can examine the trace plots to determine if the MCMC chains mixed properly. In the case of bad mixing, trace plots would typically show jumps in the graph that remain for extended periods, however, this graph does not show this issue. The density plots also look acceptable; a case of bad mixing or estimation might show density plots that are bimodal or highly skewed. While the moving average plots show peaks and valleys, the range is small ($\sim .001$), and the autocorrelation plots do not display any evidence of autocorrelation.

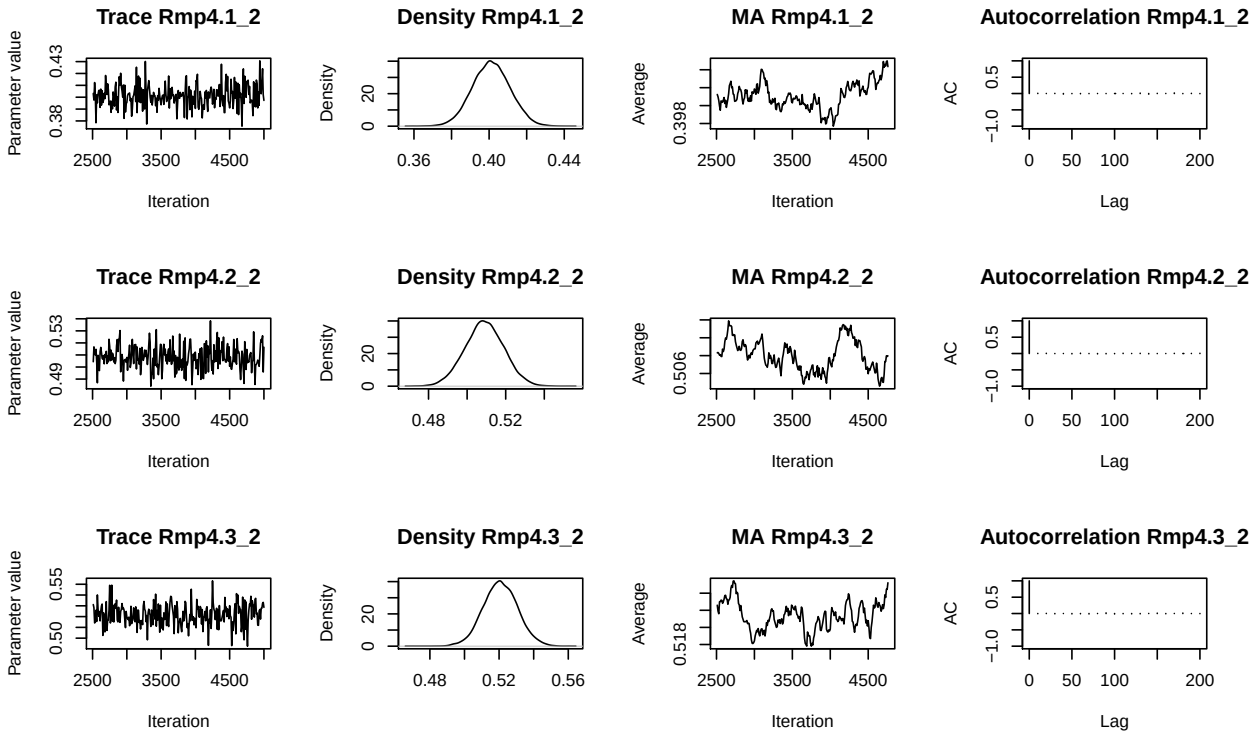

Figure 6: The graphical diagnostics for the thinned MCMC chains (with the burn-in period removed) for three of the $R_{m,p}$ parameters in the NPH model ("fit.NPH" from subsection 3.2.2), labeled as "convergenceGraphs26.pdf" in the "MRHresults_NPH" output folder . The first column contains the trace plots, the second column contains the density plot (creating using the `density()` function in R), the third column contains the moving average (with lags equal to 10% of the retained number of iterations). The fourth column contains the autocorrelation plots, calculated using the `autocorr.plot()` function in the `coda` package, which plots the autocorrelation for lags equal to 0, 1, 5, 10 and 50 times the final thinning value. This example was run for 5000 iterations, with a burn-in equal to 2500 and a thinning value equal to 10. The main purpose of these graphs is to determine if there was a mixing problem (trace plots, density plots, moving average plots) or an autocorrelation problem (the autocorrelation plot). For detailed information on interpreting graphical diagnostics, see [4, 7].

5 Other MRH Functions

After fitting the MRH model, other functions are available for use on either the fitted MRH object or on the text file containing the MCMC chains (see Section 6 for use of the text file). These functions allow the user to create and adjust graphics of the hazard rate, log hazard ratios, and aggregate measures, as well as calculate the AIC, BIC and DIC of a fitted model.

5.1 Graphics

Based on the (saved) fitted MRH object, the user can reproduce or modify the hazard rate, survival or cumulative hazard function plots. This is done by using the `plot()` command and by specifying the "plot.type" option as 'h', 'H', or 'S' for the hazard rate, cumulative hazard, or survival function, respectively. The user may specify if the credible interval bounds should be plotted as well by setting the interval option equal to `TRUE` or `FALSE`. The credible interval level is specified via the "alpha.level" option (default is equal to 0.05 for the 95% central credible interval). In addition, the graphs can be smoothed using the "smooth.graph" and "smooth.df" options. We show below the code for how this is done with our `fit.PH` example, with the corresponding plots shown in Figure 7.

```
##############################################################################
# CODE FOR PLOTTING THE FITTED MRH MODELS
##############################################################################
# Check the plot.MRH help page to see all available graphing options
?plot.MRH

# Plot the hazard rate, the cumulative hazard, and the survival function side-by-side
# with 99% credible interval bands

par(mfrow = c(1,3))
plot(fit.PH, main = 'Hazard Rate', alpha.level = 0.01)
plot(fit.PH, main = 'Cumulative Hazard', plot.type = 'H', alpha.level = 0.01)
plot(fit.PH, main = 'Survival Rate', plot.type = 'S', alpha.level = 0.01)
```

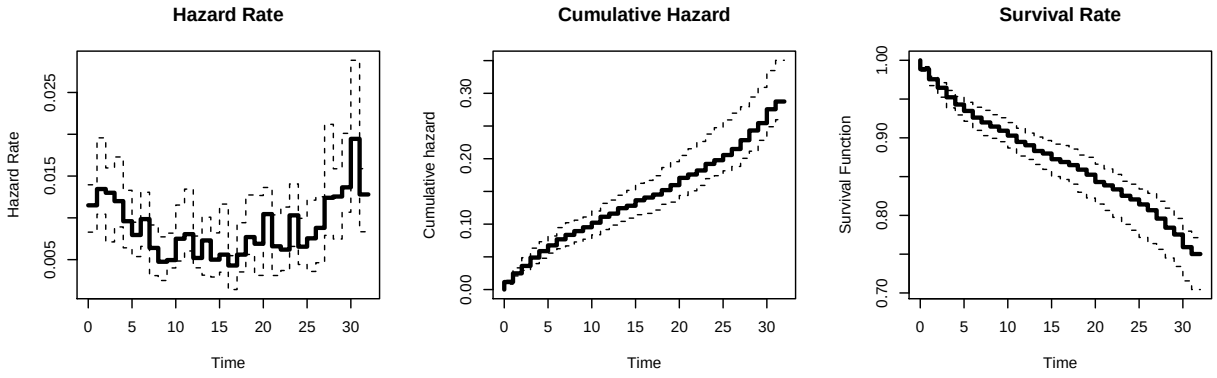

Figure 7: Using the `plot()` function on an MRH object, the user can specify the type of plot desired – posterior median hazard rate, cumulative hazard, or the survival function – by setting "plot.type" equal to 'h', 'H', or 'S', respectively. In addition, the 99% credible interval bounds are shown, obtained by setting `alpha.level = 0.01` in the `plot()` function.

The NPH model graphics are produced in the same manner, with multiple hazard rates plotted on a single graph by default. However, they may look crowded on a single graph, and in our experience usually look better if plotted

separately. This can be done by setting the "combine.graphs" option to `FALSE` in the `plot` command, as we show in the code below. To see these graphs for the `fit.NPH` example see Figure 8.

```
##############################################################################
# CODE FOR PLOTTING THE HAZARD RATES IN NON-PROPORTIONAL HAZARDS MODELS
##############################################################################

# The default plot with smoothed estimates
plot(fit.NPH, smooth.graph = TRUE)

# The smoothed estimates shown in separate plots
plot(fit.NPH, smooth.graph = TRUE, combine.graphs = FALSE)
```

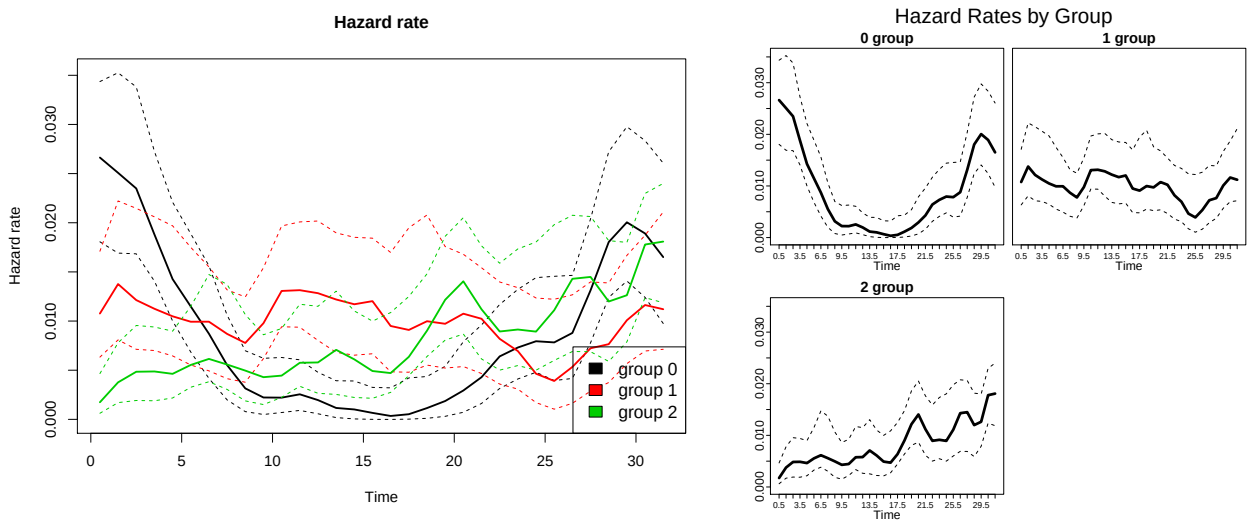

Figure 8: Using the `plot()` function on an NPH MRH object (using `fit.NPH` as an example), the user can either examine the baseline hazard rate estimate in on plot, which is the default (left), or separate them into multiple graphs by setting `combine.graphs = FALSE` in the plot command (right).

Another important feature of the NPH model is the examination of how the hazard ratios change over time. These graphs are created by setting the option `plot.type = 'r'`. The other plot settings can be controlled as described above. The results are shown in Figure 9.

```
##############################################################################
# CODE FOR PLOTTING RATIOS IN NON-PROPORTIONAL HAZARDS MODELS
##############################################################################

# Plot the log-ratios on one graph (compared to treatment group 0), omitting the credible intervals.
# Increase the level of smoothness by decreasing the degrees of freedom through "smooth.df".
plot(fit.NPH, smooth.graph = TRUE, smooth.df = 5, plot.type = 'r', interval = FALSE)

# Plot the log-ratios on separate graphs (again compared to group 0)
plot(fit.NPH, smooth.graph = TRUE, smooth.df = 5, plot.type = 'r', combine.graphs = FALSE)
```

If the user would instead like to plot the hazard ratios instead of the log-hazard ratios, the "log.ratio" option can be set to `FALSE` (default is `TRUE`).

As with the PH fitted model, the survival function and cumulative hazard function in the NPH case can also be plotted by setting "plot.type" equal to 'S' and 'H', respectively, in the `plot` command.

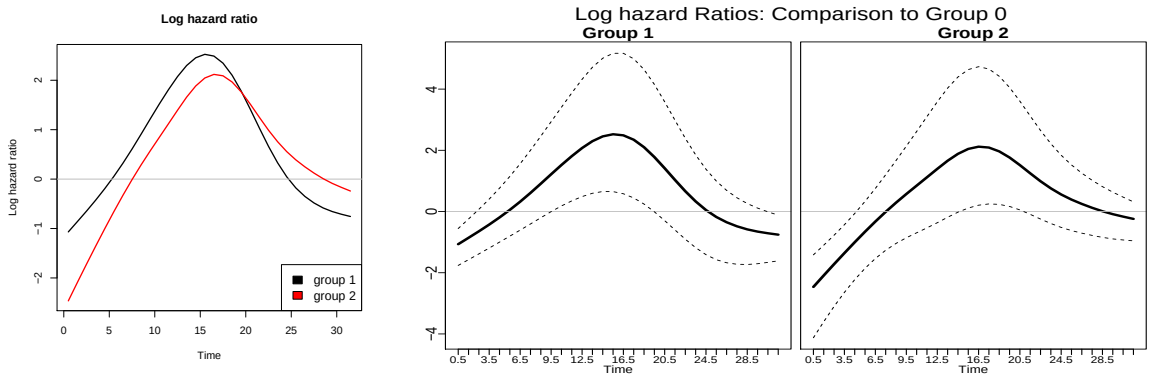

Figure 9: Using the `plot()` function on an NPH MRH object, the user can examine the log of the hazard ratios over time by setting `plot.type = 'r'`. As with the previous NPH graphs, the graphs can be combined into one graph (left, shown without credible intervals), or separated into multiple graphs (right, shown with the default 95% central credible intervals). Both graphs show smoothed posterior median estimates, with the degrees of freedom (to control smoothness) specified via the "smooth.df" option.

5.2 Model summaries

The default `summary.MRH()` function provides a subset of summaries stored in the fitted MRH object. However, there are two cases when this function is most useful. The first is to examine the credible intervals at a level other than 95% (by setting `alpha.level`). For example, to examine the hazard increments (labeled as 'd', and defined as the difference in cumulative hazard at the end and the beginning of each time bin), we type:

```
# Default summary function with 95% credible intervals
summary(fit.PH)$d
       dEst    dq.025    dq.975
d1  0.01030 0.0085975 0.0132975
...
d32 0.01295 0.0097500 0.0147625

# Summary function with the 99% credible intervals instead
summary(fit.PH, alpha.level = .01)$d
       dEst    dq.005    dq.995
d1  0.01030 0.0085195 0.0134595
...
d32 0.01295 0.0096300 0.0147925
```

The second case is when summarizing an MRH object after reading in the MCMC chains text file, as will be described in detail in Section 6).

5.3 Cumulative measures

The `summary.MRH()` function and the MRH object return the hazard increments (labeled as 'd') and the hazard rate for each bin, but not the survival or cumulative hazard functions. However, these can be obtained using the `CalcFunction()` routine. By default, the `CalcFunction()` routine returns the hazard rate, the cumulative hazard, and the survival function, with 95% credible intervals bounds included. However, the user can also specify which function is required by setting the "function.type" argument in the `CalcFunction()` to "h" (hazard rate), "H" (cumulative hazard), or "S" (survival function). The credible interval bounds can be adjusting by changing the "alpha.level" argument:

```
##############################################################################
# CODE TO CALCULATE THE SURVIVAL AND CUMULATIVE HAZARD FUNCTIONS
```

```
##############################################################################
# Default routine
nph.functions = CalcFunction(fit.NPH)
names(nph.functions)
[1] "hazrate"      "cumulhaz"     "survfunction"

# Now, let's only ask for the survival function, and change the credible interval level
surv.rate = CalcFunction(fit.NPH, function.type = 'S', alpha.level =.1)

surv.rate
$survfunction
       S.start.group1 S.bin1.group1 S.bin2.group1... S.bin30.group3   S.bin31.group3   S.bin32.group3
q.0.05              1     0.9620467     0.9456587...      0.7700274      0.7476627      0.7298485
q.0.5               1     0.9699027     0.9526894...      0.7878205      0.7664554      0.7461296
q.0.95              1     0.9777587     0.9597201...      0.8056136      0.7852481      0.7624107
```

The hazard rate and hazard increments (which in each time bin equal to the hazard rate multiplied by the length of that bin) are vectors of length 2^M. On the other hand, the survival function and the cumulative hazard function are vectors of length $2^M + 1$, as at $t = 0$, the survival function is equal 1, and the cumulative hazard is equal to 0.

5.4 Information Criteria

The Deviance Information Criterion (DIC) [10] is a Bayesian penalized goodness of fit criterion. In the MRH package, the DIC value is approximated by

$$DIC = 0.5 * var(D) + \bar{D},$$

where $D = -2 * log(L)$, and is calculated at each iteration of the MCMC chain. Lower DIC values indicate a better model fit. In addition to the DIC value, the function returns the Aikaike Information Criterion, calculated as $AIC = 2k + D$, and the Bayesian Information Criterion, calculated as $BIC = k * ln(n) + D$. In both AIC and BIC, k denotes the total number of parameters in the model. For both the AIC and BIC calculations, the maximum (i.e. "worst") value is returned to the user. In the examples shown in this vignette, these measures are useful for comparing pruned and unpruned models, and proportional hazards and non-proportional hazards models. Below is code for comparing the DIC of the three different pruned proportional hazards MRH models and the non-proportional hazards MRH model:

```
##############################################################################
# CODE FOR CALCULATING THE DIC FOR MODELS
##############################################################################

DIC(fit.PH, n = 3000)
```

This command returns the following information:

```
$loglik.summ
   Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
   7556    7566    7571    7571    7575    7589

$ICtable
       DIC      AIC      BIC
 7604.861 7661.439 7877.668
```

Similarly, we can ask for all of the information criteria directly:

```
DIC(fit.PH.prune2, n = 3000)$ICtable
       DIC      AIC      BIC
 7578.867 7605.934 7678.011
```

```
DIC(fit.PH.pruneall, n = 3000)$ICtable
       DIC      AIC      BIC
 7587.809 7605.183 7665.247

DIC(fit.NPH, n = 3000)$ICtable
       DIC      AIC      BIC
 7473.852 7633.083 8221.707
```

The DIC function returns the summary of the log likelihood values (i.e $-2 * log(lik)$) based on all kept iterations (after burn-in and thinning), as well as the DIC, AIC, and BIC values. In the above example, we compared the three MRH PH models (non-pruned, fully pruned, and partially pruned) and the MRH NPH model. We can see that among the PH models, the pruned and 2-level pruned models have very similar DIC, AIC, and BIC values, all of which are significantly lower than those of the unpruned PH model. Among all models, the DIC value is the lowest for the NPH model. However, due to the high number of parameters in the NPH model, the AIC and BIC values are higher than those of the PH models.

6 Using the MCMC Chains *.txt* File

When the `estimateMRH()` routine is run as a background job in R, the fitted object is not retained. Instead, MCMC chains are directly saved as text files in the results subfolder. For analysis, the MCMC chain text files can be read into R, and converted to an MRH object using the `as.MRH` function, and then used in the functions mentioned in previous sections.

After the chains are read in and converted to an MRH object, the only difference in the R code is that the maximum study time needs to be entered (using the "maxStudyTime" option) into some functions, as this information is not retained by the chains. Below is the code to read in the chains text file, convert it to an MRH object, and plot the results:

```
##############################################################################
# CODE TO CONVERT THE ‘‘MCMCchains.txt" FILES TO MRH OBJECTS
##############################################################################

# Read in the NPH MCMC chains file
fit.NPH.readin = as.MRH(read.table(‘MRHresults_NPH/MCMCchains.txt’, header = TRUE))
# Summarize the data
nph.summary = summary(fit.NPH.readin)

# Plot the hazards
# Note that the maximum study time must be entered or the plot function will not work
plot(fit.NPH.readin, maxStudyTime = 32)
```

7 Analyzing Multiple Chains

In MCMC sampling, often the protocol is to run multiple chains started at dispersed initial values, and then use a diagnostic that can examine evidence of stationarity based on each chain separately and combined [8]. In this instance, it is important that the MCMC chains have the same burn-in, thinning value, and number of iterations. In addition, the initial parameter starting values must be sampled from the parameter space. There are two ways that the user can satisfy these requirements using the `estimateMRH()` routine:

1. **Automated**: Setting the "GR" option equal to `TRUE` uses the automated option built in to the routine. When the user specifies "GR" equal to `TRUE`, the burn-in, maximum number of iterations, and thinning specification will be fixed to the values the user has entered (through the "burnIn", "maxIter", and "thin" options), regardless of the convergence results. The routine will also randomly sample the initial parameter values from the parameter space before starting the MCMC routine.

2. **User Specified**: The user can enter specific initial parameter values, and set the "fix.burnIn", "fix.thin", and "fix.max" option equal to `TRUE`, fixing the burn-in, maximum number of iterations, and thinning values to what has been specified by the user.

For both options, because the chain specifications (burn-in, maximum iterations, thinning value) are fixed, the user should set a `maxIter` value that is sufficiently large for convergence and analyses.

After multiple MCMC chains have been generated, the `AnalyzeMultiple()` function can be used. This function provides results for the parameters across all chains, as well as the results of the Gelman-Rubin test. The chains can be entered in to the `AnalyzeMultiple()` function as either a list of data sets or a vector of text file names:

```
##############################################################################
# CODE TO GENERATE 3 FITTED NPH MRH CHAINS AND PERFORM
# THE GELMAN-RUBIN DIAGNOSTIC TEST
##############################################################################

# Generate 3 MCMC chains for the nph model
fit.NPH1 = estimateMRH(Surv(time, delta) ~ age + gender + nph(treatment), data = MRHdata,
  M = 5, outfolder = 'MRHresults_NPH1', maxIter = 10000, GR = TRUE)
fit.NPH2 = estimateMRH(Surv(time, delta) ~ age + gender + nph(treatment), data = MRHdata,
  M = 5, outfolder = 'MRHresults_NPH2', maxIter = 10000, GR = TRUE)
fit.NPH3 = estimateMRH(Surv(time, delta) ~ age + gender + nph(treatment), data = MRHdata,
  M = 5, outfolder = 'MRHresults_NPH3', maxIter = 10000, GR = TRUE)

# The initial parameter values are different for each model, as can be checked via:
fit.NPH1$initialValues
$RmpInit
   Rmp1.0_0  Rmp2.0_0  Rmp2.1_0  ...
1 0.1514812 0.5474218 0.3064106 ...

$betaInit
        age    gender
1 0.4719683 -0.386627
...
fit.NPH2$initialValues
$RmpInit
    Rmp1.0_0  Rmp2.0_0  Rmp2.1_0  ...
1 0.02670795 0.3374994 0.9914935 ...

$betaInit
        age     gender
1 0.4574888 0.01403423
...

# Get the median estimates and 95% credible intervals for each parameter in the model,
# as well as the Gelman-Rubin test statistic.
results = AnalyzeMultiple(datalist =
  list(read.table('MRHresults_NPH1/MCMCchains.txt', header = TRUE),
  read.table('MRHresults_NPH2/MCMCchains.txt', header = TRUE),
  read.table('MRHresults_NPH3/MCMCchains.txt', header = TRUE)), maxStudyTime = 32)

# An alternative method for entering the chains, using the file names
# instead of a list containing the data sets (gives the same answer)
results = AnalyzeMultiple(fileNames = c('MRHresults_NPH1/MCMCchains.txt',
  'MRHresults_NPH2/MCMCchains.txt', 'MRHresults_NPH3/MCMCchains.txt'), maxStudyTime = 32)

names(results)
```

```
[1] "d"            "H"            "Rmp"          "beta"         "hazardrate"   "gelman.rubin"
results$gelman.rubin
            Scale Reduction Factor
beta.age                      1.00
beta.gender                   1.00
H00_0                         1.00
H00_1                         1.00
H00_2                         1.01
Rmp1.0_0                      1.00
Rmp2.0_0                      1.00
...
Rmp5.15_2                     1.00
```

For $R_{m,p}$ parameters that are pruned, the GR scale reduction factor is not calculated. A scale reduction factor that is "far" from 1 may mean that more iterations are needed, or the chain should be thinned by a larger value.

8 Summary

The MRH package offers flexible and computationally efficient set of Bayesian statistical tools for modeling survival data. While useful in a variety of settings, the package is particularly well designed for analyzing long-term survival outcomes, and provides estimates of the hazard rate and covariate effects under the proportional and non-proportional hazards assumptions. The flexible nature of this Bayesian model in conjunction with the pruning method also accommodate periods of time with sparsely observed failures.

Acknowledgements

The authors would like to acknowledge National Institute of Health grants R21 DA027624-01 and the National Science Foundation grants NSF-DEB 1316334 and NSF-GEO 1211668 that partially supported Dr. Dukic and Dr. Hagar. This work was done while Dr. Hagar was a postdoctoral researcher in the Department of Applied Mathematics at the University of Colorado at Boulder. The project utilized the Janus supercomputer, which is supported by the National Science Foundation (award number CNS-0821794) and the University of Colorado at Boulder. The Janus supercomputer is a joint effort of the University of Colorado at Boulder, the University of Colorado - Denver, and the National Center for Atmospheric Research (NCAR). Janus is operated by the University of Colorado at Boulder.

We are grateful to Steve Scott for providing advice on building an R packages. In addition, we thank the REACCTING team (www.reaccting.com) for helpful input on specific package needs for the MRH library.

References

[1] P. Bouman, J. Dignam, V. Dukic, and X.L. Meng. A multiresolution hazard model for multi-center survival studies: Application to Tamoxifen treatment in early stage breast cancer. Journal of the American Statistical Association, 102:1145–1157, 2007.

[2] P. Bouman, V. Dukic, and X.L. Meng. Bayesian multiresolution hazard model with application to an AIDS reporting delay study. Statistica Sinica, 15:325–357, 2005.

[3] Y. Chen, Y. Hagar, J. Dignam, and V. Dukic. Pruned Multiresolution Hazard (PMRH) models for time-to-event data. Bayesian Analysis, In Review.

[4] M.K. Cowless and B.P. Carlin. Markov Chain Monte Carlo convergence diagnostics: a comparative review. Journal of the American Statistical Association, 91:883–904, 1996.

[5] J. Dignam, V. Dukic, S. Anderson, E. Mamounas, D. Wickerham, and N. Wolmark. Hazard of recurrence and adjuvant treatment effects over time in lymph node-negative breast cancer. Breast Cancer Research and Treatment, 116:595–602, 2009.

[6] V. Dukic and J. Dignam. Bayesian hierarchical multiresolution hazard model for the study of time-dependent failure patterns in early stage breast cancer. Bayesian Analysis, 2:591–610, 2007.

[7] A. Gelman, J.B. Carlin, H.S. Stern, D.B. Dunson, A. Vehtari, and D.B. Rubin. Bayesian Data Analysis. CRC Press, Taylor and Francis Group, Boca Raton, FL, USA, 2014.

[8] A. Gelman and D.B. Rubin. Inference from iterative simulation using multiple sequences. Statistical Science, 7:457–511, 1992.

[9] Y. Hagar, D. Albers, R. Pivavarov, H. Chase, V. Dukic, and N. Elhadad. Survival analysis with Electronic Health Record data: Experiments with Chronic Kidney Disease. Statistical Analysis and Data Mining, to appear.

[10] D.J. Spiegelhalter, N. Best, B.P. Carlin, and A. van der Linde. Bayesian measures of model complexity and fit (with discussion). Journal of the Royal Statistical Society - Series B, 64:583–616, 2002.

Appendix A: Technical Notes

Package Dependencies

The MRH package depends on (and imports) other R packages. While almost all of the MRH package depends on custom-made functions designed specifically for our objectives, the packages listed are used for the following purposes:

- coda: The coda package provides output analysis and diagnostics for MCMC chains. The MRH package uses `autocorr.diag()` function from this package to assess the autocorrelation within the chains. http://cran.r-project.org/web/packages/coda/index.html
- HI: The HI package contains an `arms()` function that is used to perform Adaptive Rejection Metropolis Sampling (ARMS), and has been adapted for the MRH package and used in the MCMC sampling routine. http://cran.r-project.org/web/packages/HI/index.html
- KMsurv: The "tongue" data set available in KMsurv is used in the examples provided in the MRH help files for the non-proportional hazards models. http://cran.r-project.org/web/packages/KMsurv/index.html
- survival: Multiple components from the survival package are used in the MRH package. These include the survival formula format and Cox proportional hazards model estimates (used to obtain initial covariate parameter values). http://cran.r-project.org/web/packages/survival/index.html.

Because the MRH package relies on these packages, if the user does not already have them on their machine, they should be installed by setting "dependencies = TRUE" during the installation process.

More information

More information on the MRH package can be found at

http://amath.colorado.edu/faculty/vdukic/software/MRH.html.

Any questions, comments, or additional thoughts can be sent to Vanja Dukic at vanja.dukic@colorado.edu or Yolanda Hagar at yolanda.hagar@colorado.edu. We are constantly striving to improve the MRH package and appreciate any feedback on your experiences using our software extension.

Appendix B: Code from vignette

The code contained in this section is the consolidated code from the vignette.

```
# Read in the data set:
data(MRHdata)
MRHdata[1:5,]

##############################################################################
# CODE TO EXAMINE THE BIN LENGTH WITH DIFFERENT M VALUES
##############################################################################
#
# Find the bin widths for values of M ranging from 2 to 10.  Specify the time unit as weeks
# by setting "time.unit" equal to 'w'.  Other options are seconds (s), minutes (min), hours (h),
# days (d), weeks (w), months (mon), and years (y).

data(MRHdata)
FindBinWidth(MRHdata$time, MRHdata$delta, 'w')

# Re-examine the bin widths with a maximum study time equal to 32
FindBinWidth(MRHdata$time, MRHdata$delta, 'w', maxStudyTime = 32)

# Adjust the data so that the maximum study time is equal to 32
MRHdata$delta[MRHdata$time > 32] = 0
# This command changes little in the current dataset.
# Note that it can have important consequences in other datasets.
MRHdata$time[MRHdata$time > 32] = 32

###### CODE FOR FITTING THE MRH PROPORTIONAL HAZARDS MODEL ######
data(MRHdata)
fit.PH = estimateMRH(Surv(time, delta) ~ age + gender + as.factor(treatment), data = MRHdata,
M = 5, outfolder = 'MRHresultsPH', maxIter = 5000)

###### CODE FOR FITTING THE MRH NON-PROPORTIONAL HAZARDS MODEL ######
fit.NPH = estimateMRH(Surv(time, delta) ~ age + gender + nph(treatment), data = MRHdata,
M = 5, outfolder = 'MRHresults_NPH', maxIter = 5000)

##############################################################################
# CODE FOR FITTING THE PRUNED MODELS
##############################################################################

# Fit the MRH PH model, pruning all levels
#(only requires the "prune" option to be set equal to TRUE)

fit.PH.pruneall = estimateMRH(Surv(time, delta) ~ age + gender + as.factor(treatment),
data = MRHdata, M = 5, outfolder = 'MRHresults_PH_pruneall', maxIter = 5000,
prune = TRUE)


# Fit the MRH PH model, pruning only the bottom two levels
# (set "prune" equal to TRUE, and set "prune.levels" equal to 2)

fit.PH.prune2 = estimateMRH(Surv(time, delta) ~ age + gender + as.factor(treatment),
data = MRHdata, M = 5, outfolder = 'MRHresults_PH_prune2', maxIter = 5000,
prune = TRUE, prune.levels = 2)

# Information available in the MRH fitted object
```

```
names(fit.PH)
fit.PH$hazardRate

fit.NPH = estimateMRH(Surv(time, delta) ~ age + gender + nph(treatment), data = MRHdata,
M = 4, outfolder = 'MRHresults_NPH', maxIter = 1000)

##############################################################################
# CODE FOR ADJUSTING AND SAMPLING K IN THE 2-LEVEL PRUNED MODELS
##############################################################################
# Set fixed value of k equal to 0.2
fit.keq.2 = estimateMRH(Surv(time, delta) ~ age + gender + as.factor(treatment),
data = MRHdata, M = 5, outfolder = 'MRHresults_keqpoint2', maxIter = 1000,
prune = TRUE, prune.levels = 2, k.fixed = 0.2)

# Set fixed value of k equal to 0.2
fit.keq1 = estimateMRH(Surv(time, delta) ~ age + gender + as.factor(treatment),
data = MRHdata, M = 5, outfolder = 'MRHresults_keq1', maxIter = 1000,
prune = TRUE, prune.levels = 2, k.fixed = 1)

# Sample k
fit.ksample = estimateMRH(Surv(time, delta) ~ age + gender + as.factor(treatment),
                          data = MRHdata, M = 5, outfolder = 'MRHresults_ksample', maxIter = 1000,
                          prune = TRUE, prune.levels = 2, k.fixed = FALSE)

#### Same as above, but on a small subset of 300 subjects
subset = c(sample(1:500, size = 50), sample(1:500+500, size = 50),
sample(1:500+1000, size = 50), sample(1:500+1500, size = 50),
sample(1:500+2000, size = 50), sample(1:500+2500, size = 50))
# Set fixed value of k equal to 0.2
fit.keq.2_subset = estimateMRH(Surv(time, delta) ~ age + gender + as.factor(treatment),
data = MRHdata[subset,], M = 5, outfolder = 'MRHresults_keqpoint2_subset', maxIter = 1000,
prune = TRUE, prune.levels = 2, k.fixed = 0.2)

fit.keq.5_subset = estimateMRH(Surv(time, delta) ~ age + gender + as.factor(treatment),
data = MRHdata[subset,], M = 5, outfolder = 'MRHresults_keqpoint5_subset', maxIter = 1000,
prune = TRUE, prune.levels = 2)

fit.keq1_subset = estimateMRH(Surv(time, delta) ~ age + gender + as.factor(treatment),
data = MRHdata[subset,], M = 5, outfolder = 'MRHresults_keq1_subset', maxIter = 1000,
prune = TRUE, prune.levels = 2, k.fixed = 1)

fit.ksample_subset = estimateMRH(Surv(time, delta) ~ age + gender + as.factor(treatment),
data = MRHdata[subset,], M = 5, outfolder = 'MRHresults_ksample_subset', maxIter = 1000,
prune = TRUE, prune.levels = 2, k.fixed = FALSE)

##############################################################################
# CODE FOR ADJUSTING AND SAMPLING GAMMA IN THE PH and NPH MODELS
##############################################################################
# Set fixed value of gamma equal to 0.75 in PH model. Gamma is a vector with 2^M-1 bins.
fit.PH.gamma.75 = estimateMRH(Surv(time, delta) ~ age + gender + as.factor(treatment),
data = MRHdata, M = 5, outfolder = 'MRHresults_PH_gamma.75', maxIter = 1000,
gamma.fixed = rep(.75, 2^5-1))

# Set fixed value of gamma equal to 0.75 in NPH model.
# Gamma can either be entered as a vector with 2^M-1 bins, or a matrix with 2^M-1
# rows and 3 (the number of strata in the NPH treatment variable)
```

```
fit.NPH.gamma.75 = estimateMRH(Surv(time, delta) ~ age + gender + nph(treatment),
data = MRHdata, M = 4, outfolder = 'MRHresults_NPH_gamma.75', maxIter = 1000,
gamma.fixed = rep(.75, 2^4-1))
# This will do the same thing as the code above.  It's only necessary if the
# different treatment groups need different values for fixed gammas.
fit.NPH.gamma.75 = estimateMRH(Surv(time, delta) ~ age + gender + nph(treatment),
data = MRHdata, M = 4, outfolder = 'MRHresults_NPH_gamma.75', maxIter = 1000,
gamma.fixed = matrix(.75, nrow = 2^4-1, ncol = 3))

# Set fixed values in first half of the study to 0.75,
# and in the second half of the study to 0.5
fit.NPH.gamma.mixed = estimateMRH(Surv(time, delta) ~ age + gender + nph(treatment),
data = MRHdata, M = 4, outfolder = 'MRHresults_NPH_gamma.mixed', maxIter = 1000,
gamma.fixed = c(rep(0.75, 8), rep(0.5, 7)))

# Sample the gamma values (estimate based on the observed data) for the PH model
fit.PH.gamma.sampled = estimateMRH(Surv(time, delta) ~ age + gender + as.factor(treatment),
data = MRHdata, M = 4, outfolder = 'MRHresults_PH_gamma.sampled', maxIter = 1000,
gamma.fixed = FALSE)

##############################################################################
# CODE FOR PLOTTING THE FITTED MRH MODELS
##############################################################################
# Check the plot.MRH help page to see all available graphing options
?plot.MRH
# Plot the hazard rate, the cumulative hazard, and the survival function side-by-side
par(mfrow = c(1,3))
plot(fit.PH, main = 'Hazard Rate', alpha.level = 0.01)
plot(fit.PH, main = 'Cumulative Hazard', plot.type = 'H', alpha.level = 0.01)
plot(fit.PH, main = 'Survival Rate', plot.type = 'S', alpha.level = 0.01)

##############################################################################
# CODE FOR PLOTTING THE HAZARD RATES IN NON-PROPORTIONAL HAZARDS MODELS
##############################################################################

# The default plot with smoothed estimates
plot(fit.NPH, smooth.graph = TRUE)
# The new plot, with smoothed estimates in separate graphs
plot(fit.NPH, smooth.graph = TRUE, combine.graphs = FALSE)

##############################################################################
# CODE FOR PLOTTING RATIOS IN NON-PROPORTIONAL HAZARDS MODELS
##############################################################################

# Plot the log-ratios on one graph (compared to treatment group 0), omitting the credible intervals.
# Increase the level of smoothness by decreasing the degrees of freedom through "smooth.df".
plot(fit.NPH, smooth.graph = TRUE, smooth.df = 5, plot.type = 'r', interval = FALSE)
# Plot the log-ratios on separate graphs (again compared to group 0)
plot(fit.NPH, smooth.graph = TRUE, smooth.df = 5, plot.type = 'r', combine.graphs = FALSE)

# Default summary function with 95% credible intervals
summary(fit.PH)$d
# Examine the 99% credible intervals instead
summary(fit.PH, alpha.level = .01)$d

##############################################################################
```

```
# CODE TO CALCULATE THE SURVIVAL AND CUMULATIVE HAZARD FUNCTIONS
################################################################################
# Default routine
nph.functions = CalcFunction(fit.NPH)
names(nph.functions)
surv.rate = CalcFunction(fit.NPH, function.type = 'S', alpha.level =.1)
surv.rate

################################################################################
# CODE FOR CALCULATING THE DIC FOR MODELS
################################################################################

DIC(fit.PH, n = 3000)
DIC(fit.PH.prune2, n = 3000)$ICtable
DIC(fit.NPH, n = 3000)$ICtable

################################################################################
# CODE TO CONVERT THE ''MCMCchains.txt" FILES TO MRH OBJECTS
################################################################################

# Read in the NPH MCMC chains file
fit.NPH.readin = as.MRH(read.table('MRHresults_NPH/MCMCchains.txt', header = TRUE))
# Summarize the data
nph.summary = summary(fit.NPH.readin)

# Plot the hazards
# Note that the maximum study time must be entered or the plot function will not work
plot(fit.NPH.readin)
plot(fit.NPH.readin, maxStudyTime = 32)

################################################################################
# CODE TO GENERATE 3 FITTED MRH CHAINS AND PERFORM
# THE GELMAN-RUBIN DIAGNOSTIC TEST
################################################################################

# Generate 3 MCMC chains for the nph model
fit.NPH1 = estimateMRH(Surv(time, delta) ~ age + gender + nph(treatment), data = MRHdata,
M = 5, outfolder = 'MRHresults_NPH1', maxIter = 10000, GR = TRUE)
fit.NPH2 = estimateMRH(Surv(time, delta) ~ age + gender + nph(treatment), data = MRHdata,
M = 5, outfolder = 'MRHresults_NPH2', maxIter = 10000, GR = TRUE)
fit.NPH3 = estimateMRH(Surv(time, delta) ~ age + gender + nph(treatment), data = MRHdata,
M = 5, outfolder = 'MRHresults_NPH3', maxIter = 10000, GR = TRUE)

# The initialized parameter values are different for each model
fit.NPH1$initialValues
fit.NPH2$initialValues

# Get the median estimates and 95% credible intervals for each parameter in the model,
# as well as the Gelman-Rubin test statistic.
results = AnalyzeMultiple(datalist =
list(read.table('MRHresults_NPH1/MCMCchains.txt', header = TRUE),
read.table('MRHresults_NPH2/MCMCchains.txt', header = TRUE),
read.table('MRHresults_NPH3/MCMCchains.txt', header = TRUE)), maxStudyTime = 32)

# An alternative method for entering the data sets, using the file names
# instead of a list containing the data sets (gives the same answer)
```

```
results = AnalyzeMultiple(fileNames = c('MRHresults_NPH1/MCMCchains.txt',
'MRHresults_NPH2/MCMCchains.txt', 'MRHresults_NPH3/MCMCchains.txt'), maxStudyTime = 32)
names(results)
```