Using **icenReg** for interval censored data in **R** v2.0.9

Clifford Anderson-Bergman

October 3, 2020

Contents

1 Introduction

This manual is meant to provide an introduction to using **icenReg** to analyze interval censored data. It is written with expectation that the reader is familiar with basic survival analysis methods. Familiarity with the Kaplan Meier curves, survival regression models (Cox PH, AFT and proportional odds) and basic Bayesian principles should be sufficient for all that is covered in this text.

1.1 Interval Censoring

Interval censoring occurs when a response is known only up to an interval. A classic example is testing for diseases at a doctor's clinic; if a subject tests negative at t_1 and positive at t_2, all that is known is that the subject acquired the disease in (t_1, t_2), rather than an exact time. Other classic examples include examining test mice for tumors after sacrifice (results in *current status* or

case I interval censored data, in which all observations are either left or right censored, as opposed to the more general *case II*, which allows for any interval), customer choice models in economics (customers are presented a price for a product and chose to purchase or not, researcher wants to know distribution of maximum spending amount; this results in current status data again), data reduction methods for sensor analyses (to reduce load on sensor system, message is intentionally surpressed if outcome is in an expected region) and data binning (responses reported only up to an interval, in some cases to keep the subjects anonymous, in some cases to reduce size of data).

Often interval censoring is ignored in analysis. For example, age is usually reported only up to the year, rather than as a continuous variable; when a subject reports that their age is 33, the information we have is really that their age is in the interval [33,34). In the case that these intervals are very short relative to the question of interest, such as with reported age when the scientific quesiton is about age of onset of type II diabetes, the bias introduced by ignoring the interval censoring may besmall enough to be safely ignored. However, in the case that the width of intervals is non-trivial, statistical methods that account for this should be used for reliable analysis.

Standard notation for interval censoring is that each subject has a true event time t_i which is potentially unobserved. Rather, we observe a response interval l_i, r_i such that the true event time is known to have occurred within. Default behavior is to assume this an half open, half closed interval, i.e. $t_i \in (l_i, r_i]$[1]. An exception to the rule in this notation is that $(t_i, t_i]$ always implies that event i is uncensored with event time at t_i, despite this being an undefined interval. This allows for uncensored observations ($l_i = r_i$), right censored($r_i = \infty$), left censored ($l_i = 0$) or none of the above ($0 < l_i < r_i < \infty$).

In **icenReg**, several models are included in which the response value is allowed to be interval censored. If our data contains the values `L` and `R`, representing the left and right sides of the response interval, we can pass our response to a regression model using either

```
cbind(L, R)
Surv(L, R, type = "interval2")
```

1.2 Classic Estimators

The topic of interval censoring began in the field of survival analysis. Although it is now considered in other fields of study (such as tobit regression), at this time **icenReg** focusses on survival models.

One of the earliest models is the Non-Parametric Maximum Likelihood Estimator (NPMLE), also referred to as Turnbull's Estimator. This is a generalization of the Kaplan Meier curves (which is a generalization of the empirical distribution function) that allows for interval censoring. Unlike the Kaplan Meier curves, the solution is not in closed form and several algorithms have been proposed for efficient computation.

Semi-parametric models exist in the literature as well; two classic regression models fit by **icenReg** are the Cox-PH model and the proportional odds model.

[1] In **icenReg**, whether the boundaries of the interval are included or not can be controlled by the option `B` in the model fitting functions.

The well known Cox-PH, or proportional hazards regression model, has the property that

$$h(t|X,\beta) = h_o(t)e^{X^T\beta}$$

where $h(t|X,\beta)$ is the hazard rate conditional on covariates X and regression parameters β, with h_o as the baseline hazard function. This relation is equivalent to

$$S(t|X,\beta) = S_o(t)^{e^{X^T\beta}}$$

where $S(t|X,\beta)$ is the conditional survival and $S_o(t)$ is the baseline survival function.

The somewhat less common proportional odds model can be expressed as

$$\text{Odds}(S(t|X,\beta)) = e^{X^T\beta}\text{Odds}(S_o(t))$$

or

$$\frac{S(t|X,\beta)}{1-S(t|X,\beta)} = e^{X^T\beta}\frac{S_o(t)}{1-S_o(t)}$$

Unlike the special example of the Cox PH model with right-censored data, it is very difficult to estimate the regression parameters without estimating the baseline distribution[2]. The model can be kept semi-parametric (i.e. no need to decide on a parametric baseline distribution) by using the Turnbull estimator, modified to account for the given regression model, as the baseline distribution. The semi-parametric model can be computationally very difficult, as the number of baseline parameters can be quite high (up to n), which must follow shape constraints (i.e. either a set of probability masses or a cumulative hazard function, which must be strictly increasing) and there is no closed form solution to either regression or baseline parameters. One of the contribution the algorithms in **icenReg** make to the field of statistical computing is efficient computation of the non-parametric and semi-parametric estimators, allowing for relatively efficient estimation on standard computers (*i.e.* less than one second) of relatively large samples (n = 10,000 for the semi-paramtric model, n = 100,000 for the non-parametric model), although the semi-parametric models are still significantly slower than fully-parametric models.

Fully parametric models exist as well and can be calculated using fairly standard algorithms. In addition to the proportional hazards and odds models, the accelerated failure time model can be used for parameteric modeling. These models have the following relationship:

$$S(t|X,\beta) = S_o(te^{X^T\beta})$$

For technical reasons not discussed here, this model is very simple to implement for a fully parameteric model, but very difficult for a semi-parametric model. As such, **icenReg** contains tools for a fully-parametric accelerated failure time model, but not a semi-parametric one.

There are slight complications in that the interval censoring can cause the log likelihood function to be non-concave. However, for reasonable sized data, the

[2]Methods to estimate the regression parameters without estimating the baseline parameters have been proposed, but at great computational cost without any clear statistical benefit.

log likelihood function is usually locally concave near the mode and only slight modifications are required to address this issue. In practice, fully-parametric models should be used with caution; the lack of observed values means that model inspection can be quite difficult; there are no histograms, etc., to be made. As such, even if fully parametric models are to be used for the final analysis, it is strongly encouraged to use semi-parametric models at least for model inspection. **icenReg** fits fully parametric accelerated failure time, proportional odds and proporitonal hazard models for interval censored data.

1.3 Models fit with icenReg

At this time, the following set of models can be fit (name in paratheses is function call in **icenReg**):

- NPMLE (`ic_np`)
- Semi-parametric model (`ic_sp`)
  - `model` for model type
    - `"po"` for proportional odds
    - `"ph"` for proportional hazards
- Fully parametric frequentest model (`ic_par`) or Bayesian model (`ic_bayes`)
  - `model` for model type
    - `"po"` for proportional odds
    - `"ph"` for proportional hazards
    - `"aft"` for accelerated failure time model
  - `dist` for baseline distribution
    - `"exponential"`
    - `"gamma"`
    - `"weibull"`
    - `"lnorm"`
    - `"loglogistic"`
    - `"generalgamma"`

In addition, **icenReg** includes various diagnostic tools. These include

- Plots for diagnosising baseline distribution (`diag_baseline`)
- Plots for diagnosising covariate effects (`diag_covar`)

1.4 Data Examples in icenReg

The package includes 3 sources of example data: one function that simulates data and two sample data sets. The simulation function is `simIC_weib`, which simulates interval censored regression data with a Weibull baseline distribution The sample data sets are `miceData`, which contains current status data regarding lung tumors from two groups of mice and `IR_diabetes`, which includes data on time from diabetes to diabetic nephronpathy in which 136 of 731 observations are interval censored due to missed follow up.

2 Fitting Models using icenReg

2.1 Non-parametric models

The non-parametric maximum likelihood estimator (NPMLE) can be fit using `ic_np`. If the data set is relatively small and the user is interested in non-parametric tests, such as the log-rank statistic, we actually advise using the **interval** package, as this provides several testing functions. However, computing the NPMLE **icenReg** is several fold faster than **interval**, so if large datasets are used (i.e. n > 1,000), the user may have no choice but to use **icenReg** to fit the NPMLE. And while hypothesis testing is not currently available for the NPMLE in **icenReg**, these fits can still provide visual diagnostics for semi-parametric and fully parametric models.

To fit an NPMLE model for interval censored data, we will consider the `miceData` provided in **icenReg**. This dataset contains three variables: `l, u` and `grp`. `l` and `u` represent the left and right side of the interval containing the event time (note: data is current status) and `grp` is a group indicator with two categories.

```
> data(miceData)
> head(miceData, 3)

  l   u grp
1 0 381  ce
2 0 477  ce
3 0 485  ce
```

We can fit a non-parametric estimator for each group by

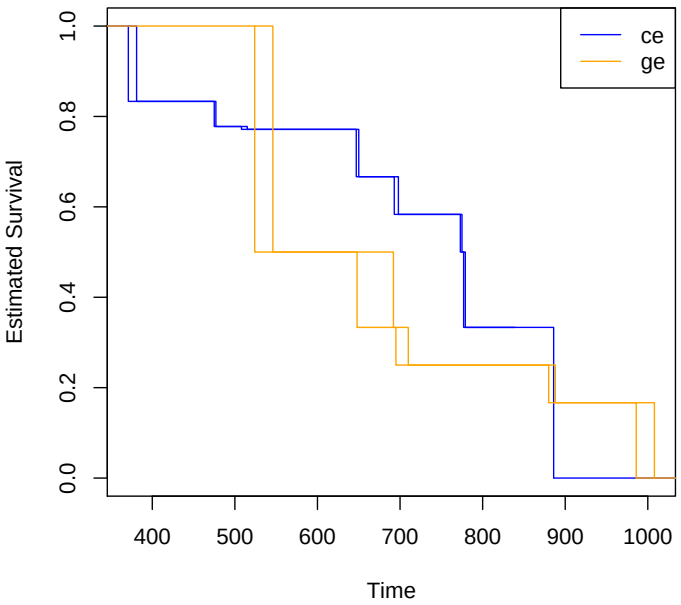
```
> np_fit = ic_np(cbind(l, u) ~ grp, data = miceData)
```

If we wanted only a single fit for both groups, this can be done in two ways. The two fits are equivalent, but are just used to demonstrate differing possible syntax.

```
> groupedFit1 <- ic_np(cbind(l,u) ~ 0, data = miceData)
> groupedFit2 <- ic_np(miceData[,c('l', 'u')])
```

The fits can be plotted as follows:

```
> plot(np_fit, col = c('blue', 'orange'),
+      xlab = 'Time', ylab = 'Estimated Survival')
```



Looking at the plots, we can see a unique feature about the NPMLE for interval censored data. That is, there are *two* lines used to represent the survival curve. This is because with interval censored data, the NPMLE is not always unique; any curve that lies between the two lines has the same likelihood. For example, any curve that lies between the two blues lines maximizes the likelihood associated with `"ge"` group of mice.

2.2 Semi-parametric models

Semi-parametric models can be fit with `ic_sp` function. This function follows standard regression syntax. As an example, we will fit the `IR_diabetes` dataset, which contains data on time from diabetes to diabetic nephropathy. In this dataset, we have the left and right sides of the observation interval containing the true response time and the gender of the patient.

```
> data("IR_diabetes")
> head(IR_diabetes, 3)

  left right gender
1   24    27   male
2   22    22 female
3   37    39   male
```

We fit the model below. Note that this may be time consuming, as the semi-parametric model is somewhat computationally intense and we are taking `bs_samples` bootstrap samples of the estimator.

```
>   fit_ph <- ic_sp(cbind(left, right) ~ gender, model = 'ph',
+                   bs_samples = 100, data = IR_diabetes)
>   fit_po <- ic_sp(cbind(left, right) ~ gender, model = 'po',
+                   bs_samples = 100, data = IR_diabetes)
```

The first model by default fits a Cox-PH model, while the second fits a proportional odds model, as controlled by the `model` argument. We can look at the results using either the `summary` function, or just directly looking at the results (what is displayed is the same).

```
>   fit_po

Model:  Proportional Odds
Dependency structure assumed: Independence
Baseline:  semi-parametric
Call: ic_sp(formula = cbind(left, right) ~ gender, data = IR_diabetes,
    model = "po", bs_samples = 100)

           Estimate Exp(Est) Std.Error z-value        p
gendermale   0.4013    1.494    0.1405   2.856 0.004286

final llk =  -1962.4
Iterations =  25
Bootstrap Samples =  100

>   fit_ph

Model:  Cox PH
Dependency structure assumed: Independence
Baseline:  semi-parametric
Call: ic_sp(formula = cbind(left, right) ~ gender, data = IR_diabetes,
    model = "ph", bs_samples = 100)

           Estimate Exp(Est) Std.Error z-value       p
gendermale  -0.1402   0.8692   0.07168  -1.956 0.05042

final llk =  -1964.96
Iterations =  38
Bootstrap Samples =  100
```

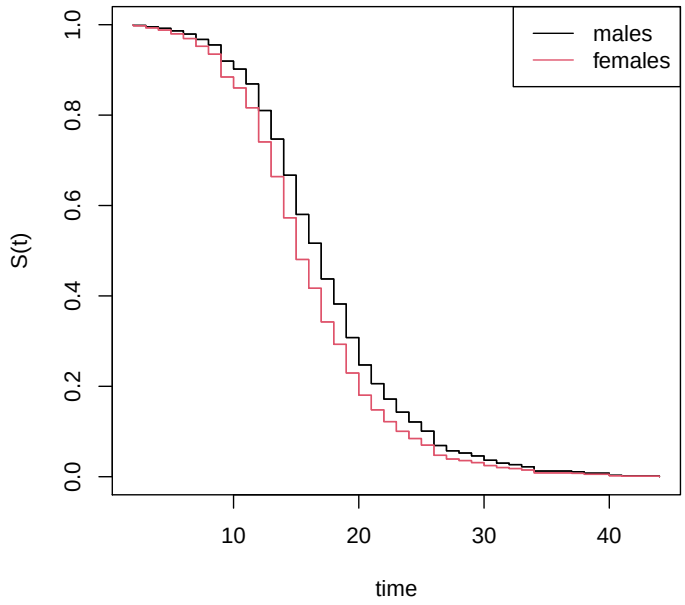
For the semi-parametric models, bootstrap samples are used for inference on the regression parameters. The reason for this is that as far as we know, the limiting distribution of the baseline distribution is currently not characterized. In fact, to our knowledge, even using the bootstrap error estimates for the baseline distribution is not valid. Because the regression parameters cannot be seperated in the likelihood function, using the negative inverse of the Hessian for the regression standard errors is not generally valid. However, it has been shown that using the bootstrap for inference *on the regression parameters* leads to valid inference.

We can use these fits to create plots as well. The `plot` function will plot the estimated survival curves or CDF for subjects with the set of covariates provided

in the `newdata` argument. If `newdata` is left equal to `NULL`, the baseline survival function will be plotted.

Below is a demonstration of how to plot the semi-parametric fit for males and females.

```
>   newdata <- data.frame(gender = c('male', 'female') )
>   rownames(newdata) <- c('males', 'females')
>   plot(fit_po, newdata)
```



2.3 Parametric Models

We can fit parametric models in **icenReg** using the `ic_par` function. The syntax is essentially the same as above, except that the user needs to specify `dist`, the parametric family that the baseline distribution belongs to. The current choices are `"exponential"`, `"weibull"` (default), `"gamma"`, `"lnorm"` (log-normal), `"loglogistic"` and `"generalgamma"` (generalized gamma distribution). The user must also select `model = "ph"`, `"po"`, or `"aft"` as the model type.

It is not necessary to specify `bs_samples` for parametric models, as inference is done using the asymptotic normality of the estimators. Fitting a parametric model is typically faster than the semi-parametric model, even if no bootstrap samples are taken for the semi-parametric model. This is because the fully-parametric model is of lower dimensional space without constraints.

Suppose we wanted to fit a proportional odds model to the `IR_diabets` data with a baseline gamma distribution. This could be fit by

```
> fit_po_gamma <- ic_par(cbind(left, right) ~ gender,
+     data = IR_diabetes, model = "po", dist = "gamma")
```

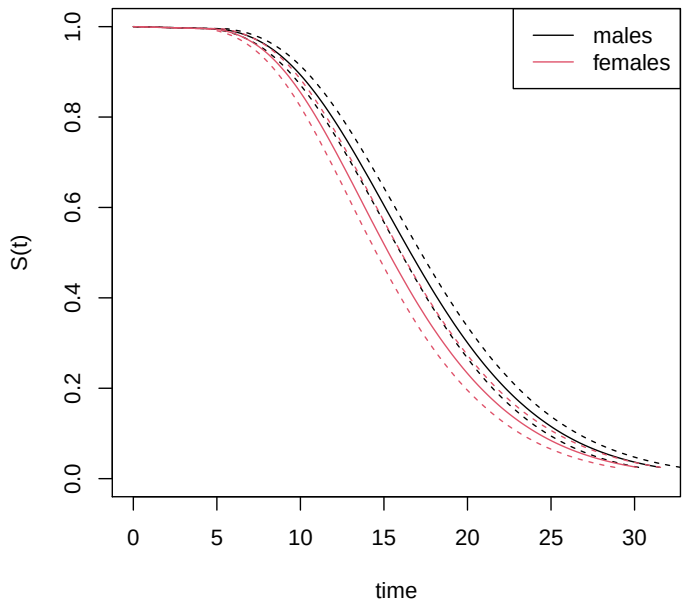
We can examine the regression coefficients in the same way as with the semi-parametric model.

```
> fit_po_gamma
```

```
Model:  Proportional Odds
Dependency structure assumed: Independence
Baseline:  gamma
Call: ic_par(formula = cbind(left, right) ~ gender, data = IR_diabetes,
    model = "po", dist = "gamma")

            Estimate Exp(Est) Std.Error z-value        p
log_shape     1.9980    7.377   0.05450  36.670 0.000000
log_scale     0.8248    2.281   0.05563  14.830 0.000000
gendermale    0.3496    1.419   0.13540   2.582 0.009824

final llk =  -2006.619
Iterations =  4
```

We can also examine the survival/cdf plots in the same way.

```
> plot(fit_po_gamma, newdata, lgdLocation = "topright")
```



2.4 Bayesian Parametric Models

The option for parametric Bayesian models is also permitted in **icenReg** with the `ic_bayes` function. If a flat prior is desired, the use is exactly identical to `ic_par`.

```
> flatPrior_fit <- ic_bayes(cbind(left, right) ~ gender,
+     data = IR_diabetes, model = "po", dist = "gamma")
```

The standard methods work in the same manner as other **icenReg** fits, although the basic output looks fairly different.

```
> flatPrior_fit

Model:  Bayesian Proportional Odds
Dependency structure assumed: Independence
Baseline:  gamma
Call: ic_bayes(formula = cbind(left, right) ~ gender, data = IR_diabetes,
    model = "po", dist = "gamma")


Iterations = 2001:6996
Thinning interval = 5
Number of chains = 4
Sample size per chain = 1000

1. Empirical mean and standard deviation for each variable,
   plus standard error of the mean:

             Mean      SD  Naive SE Time-series SE
log_shape  1.9957 0.05514 0.0008718       0.001387
log_scale  0.8273 0.05642 0.0008921       0.001365
gendermale 0.3443 0.13852 0.0021901       0.003549

2. Quantiles for each variable:

              2.5%    25%    50%    75%  97.5%
log_shape  1.88703 1.9592 1.9957 2.0325 2.1073
log_scale  0.71722 0.7898 0.8278 0.8638 0.9399
gendermale 0.07498 0.2502 0.3427 0.4392 0.6181

3. MAP estimates:
 log_shape  log_scale gendermale
    1.9640     0.8825     0.3811
```

User that want access to the raw samples can do so through the `$samples` field.

```
> head(flatPrior_fit$samples)

     log_shape log_scale gendermale
[1,]  1.964228 0.8824987  0.3810635
```

```
[2,]  1.952654 0.8871974  0.5133762
[3,]  1.934548 0.9154109  0.3728349
[4,]  1.989661 0.8314183  0.4219186
[5,]  1.887253 0.9348310  0.2924901
[6,]  1.970945 0.8458227  0.2371028
```

This object is a matrix of samples from all chains combined. Alternatively, one can look at the samples from each chain individually through the `$mcmcList` field, with each element corresponding to a given chain. Note that `$mcmcList` is a `coda` object of class `mcmc.list`.

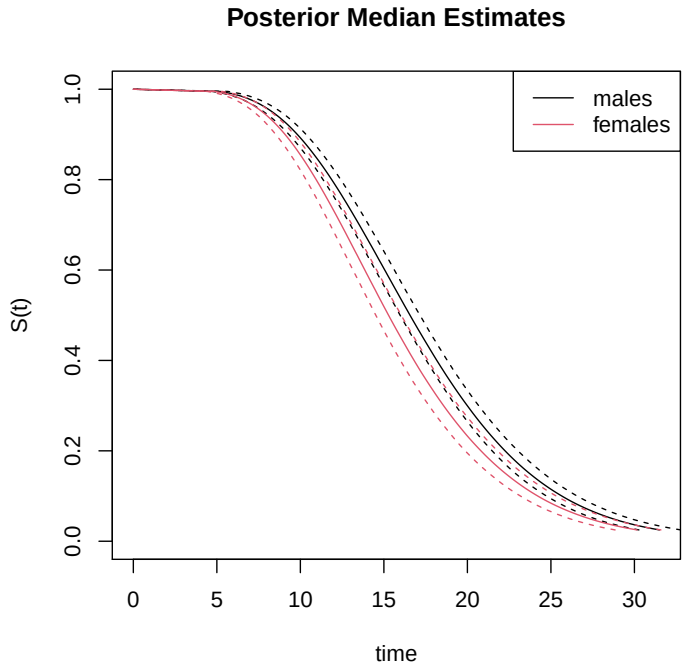
```
> # Accessing the first few samples of the first chain
> head(flatPrior_fit$mcmcList[[1]])

Markov Chain Monte Carlo (MCMC) output:
Start = 2001
End = 2031
Thinning interval = 5
     log_shape log_scale gendermale
[1,]  1.964228 0.8824987  0.3810635
[2,]  1.952654 0.8871974  0.5133762
[3,]  1.934548 0.9154109  0.3728349
[4,]  1.989661 0.8314183  0.4219186
[5,]  1.887253 0.9348310  0.2924901
[6,]  1.970945 0.8458227  0.2371028
[7,]  1.901528 0.9175980  0.3802026

> # Accessing the first few samples of the second chain
> head(flatPrior_fit$mcmcList[[2]])

Markov Chain Monte Carlo (MCMC) output:
Start = 2001
End = 2031
Thinning interval = 5
     log_shape log_scale gendermale
[1,]  1.889634 0.9214509 0.16767192
[2,]  2.109365 0.6974102 0.40372454
[3,]  2.030917 0.7856057 0.06866599
[4,]  1.932695 0.8847328 0.27433798
[5,]  1.982894 0.8397385 0.03265403
[6,]  2.008350 0.8292230 0.08944306
[7,]  2.038894 0.7929645 0.18202894
```

The log posterior density at each iteration can be accessed through the `$logPosteriorDensities` field. Note that this is a list, with each item of the list being a vector from each chain.

```
> head(flatPrior_fit$logPosteriorDensities[[1]])

[1] -2008.264 -2008.548 -2009.136 -2006.791 -2008.665 -2007.161
```

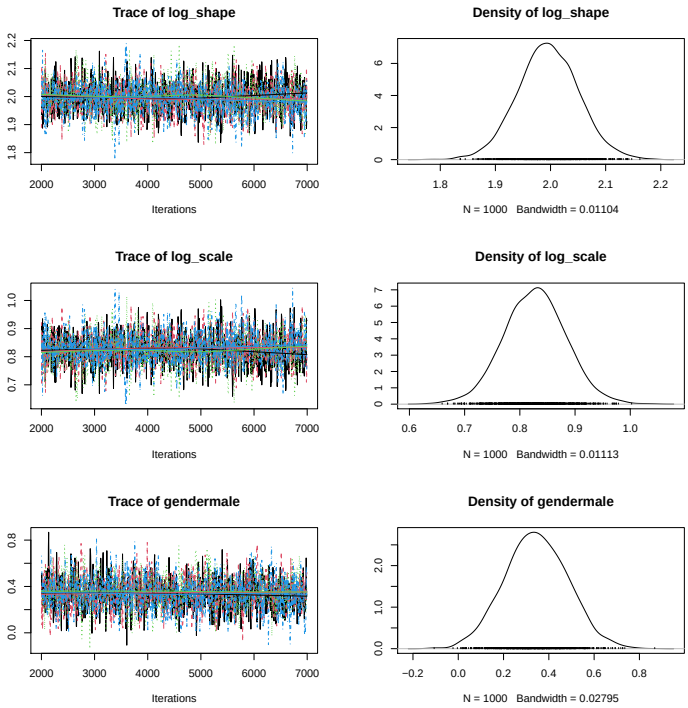
Plotting calls are as standard. Important note: the solids plots the posterior median estimates of the survival curve for a subject with given covariates, *not* the posterior mean estimates!

```
>   plot(flatPrior_fit, newdata,
+        main = 'Posterior Median Estimates')
```



MCMC diagnositic plots can be generated by accessing the `$mcmcList` field.

```
>   plot(flatPrior_fit$mcmcList)
```



As noted earlier, default behavior is to use a flat prior. If a user wants to use an informative prior, they can provide whichever custom written prior they choose. This function must take in a vector of values corresponding to all the parameters of the model (i.e. first baseline parameters and then regression parameters) and return the log-prior, calculated up to a additive constant.

This is illustrated with an example where we decide to put a tight prior about the shape parameter (i.e. first baseline parameter) and the regression parameter, and a diffuse prior about the scale parameter.

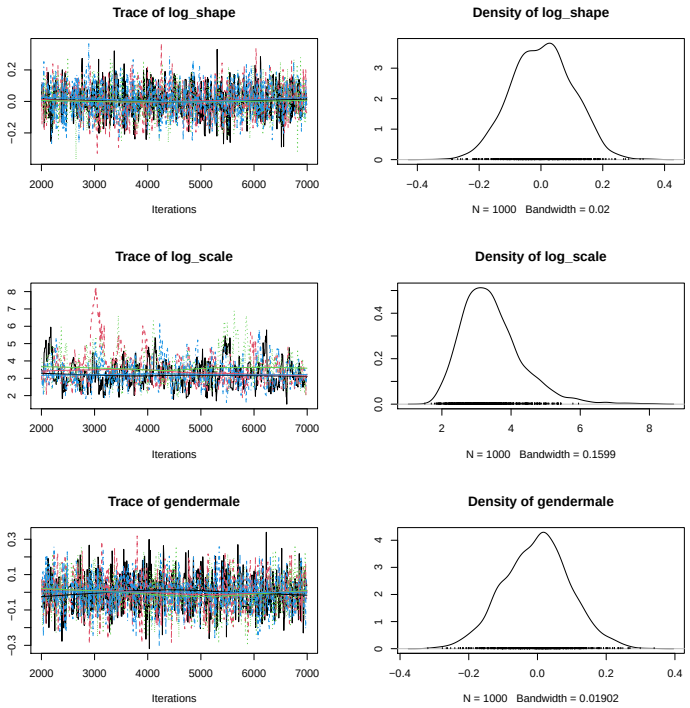
```
>  logPriorFunction <- function(x){
+    ans <- 0
+    ans <- ans + dnorm(x[1], sd = 0.1, log = T)
+    # Tight prior about 1st parameter, log_shape
+    ans <- ans + dnorm(x[2], sd = 10, log = T)
+    # Diffuse prior about 2nd parameter, log_scale
+    ans <- ans + dnorm(x[3], sd = 0.1, log = T)
+    # Tight prior about 3rd parameter, regression parameter
+    return(ans)
+  }
> informPrior_fit <- ic_bayes(cbind(left, right) ~ gender,
+     data = IR_diabetes, model = "po", dist = "gamma",
+     logPriorFxn = logPriorFunction)
> # Fitting model with prior.
>
> informPrior_fit

Model:  Bayesian Proportional Odds
```

```
Dependency structure assumed: Independence
Baseline:  gamma
Call: ic_bayes(formula = cbind(left, right) ~ gender, data = IR_diabetes,
    logPriorFxn = logPriorFunction, model = "po", dist = "gamma")


Iterations = 2001:6996
Thinning interval = 5
Number of chains = 4
Sample size per chain = 1000

1. Empirical mean and standard deviation for each variable,
   plus standard error of the mean:

              Mean      SD  Naive SE Time-series SE
log_shape  1.45195 0.05831 0.0009220       0.001589
log_scale  1.36460 0.06028 0.0009532       0.001676
gendermale 0.09693 0.07770 0.0012286       0.001614

2. Quantiles for each variable:

               2.5%     25%     50%    75%  97.5%
log_shape   1.34163 1.41277 1.45135 1.4925 1.5657
log_scale   1.24569 1.32366 1.36556 1.4049 1.4818
gendermale -0.05768 0.04365 0.09785 0.1496 0.2473

3. MAP estimates:
 log_shape  log_scale gendermale
    1.4720     1.3790     0.1072
```

Note that the posterior mean for `log_shape` is pulled down significantly.

Default behavior for `ic_bayes` is to use an adaptive MH block updater, started from the MLE (not MAP) estimate to seed the starting point of the chain and starting proposal covariance. The proposal covariance is updated as more samples are taken. This should work well with informative likelihoods and relatively weak priors. On the other hand, it can be quite problematic with weak likelihoods and relatively strong priors. In such cases, a user may prefer to turn off the MLE initialization. This can be done with the `bayesControl` function.

```
> weak_data <- IR_diabetes[1:2,]
> weakData_fit <- ic_bayes(cbind(left, right) ~ gender,
+     data = weak_data,
+     model = "po", dist = "gamma",
+     logPriorFxn = logPriorFunction,
+     controls = bayesControls(useMLE_start = F))
> plot(weakData_fit$mcmcList)
```



Note that in this sample, we see evidence that the chain has not properly mixed, especially in regards to the `log_scale` parameter. Using `bayesControls`, we could increase both the samples and burnIn to address this issue.

2.5 Extracting Estimates and CIs

We can also extract survival estimates and confidence/credible intervals for a given set of covariates using the `survCIs`.

```
> # Extract estimates for inverse CDF
> invCDF_ests = survCIs(informPrior_fit, newdata,
+                       p = seq(from = 0.05, to = .95, by = 0.2))
> # Extract estimates for *CDF* probabilities at given values
> CDF_ests = survCIs(informPrior_fit, newdata,
+                    q = seq(from = 5, to = 25, by = 5))
> invCDF_ests

Model call:
  ic_bayes(formula = cbind(left, right) ~ gender, data = IR_diabetes,
    logPriorFxn = logPriorFunction, model = "po", dist = "gamma")
Credible Level = 0.95
Rowname:  males
     Percentile estimate (mean) estimate (median)     lower     upper
[1,]       0.05        6.039777          6.043296  5.529854  6.570387
[2,]       0.25       10.932574         10.937775 10.366248 11.497208
[3,]       0.45       14.655778         14.666036 14.046838 15.268631
[4,]       0.65       18.802747         18.805955 18.080582 19.534243
```

```
[5,]       0.85       25.100311         25.096739 24.082415 26.144175
Rowname:  females
     Percentile estimate (mean) estimate (median)     lower     upper
[1,]       0.05        5.863795           5.86078  5.338848  6.400758
[2,]       0.25       10.589113          10.58836  9.957947 11.213920
[3,]       0.45       14.210123          14.21114 13.499419 14.929923
[4,]       0.65       18.284241          18.28183 17.479092 19.130894
[5,]       0.85       24.540875          24.52850 23.488323 25.717288
```

```
> CDF_ests
```

```
Model call:
  ic_bayes(formula = cbind(left, right) ~ gender, data = IR_diabetes,
    logPriorFxn = logPriorFunction, model = "po", dist = "gamma")
Credible Level = 0.95
Rowname:  males
     Time estimate (mean) estimate (median)      lower      upper
[1,]    5      0.02764125         0.0273352 0.01928434 0.03721722
[2,]   10      0.20305819         0.2025993 0.17624982 0.23107802
[3,]   15      0.46824599         0.4676999 0.43616014 0.50057971
[4,]   20      0.69812680         0.6981342 0.66854188 0.72736433
[5,]   25      0.84774364         0.8478515 0.82454256 0.87004162
Rowname:  females
     Time estimate (mean) estimate (median)      lower      upper
[1,]    5      0.03037879        0.03015163 0.02128665 0.04119292
[2,]   10      0.21924190        0.21908613 0.18886955 0.25222601
[3,]   15      0.49241230        0.49216053 0.45375121 0.53060664
[4,]   20      0.71805259        0.71831844 0.68435947 0.74906223
[5,]   25      0.85973632        0.86013107 0.83461798 0.88199072
```

Note that the output from survCIs is what is used to create the plots for `icenReg` fits, so if one wanted to export the plotting to `ggplot` or plot the posterior mean rather than the posterior median, for example, one could do so directly with `survCIs`.

3 Inspecting model fit

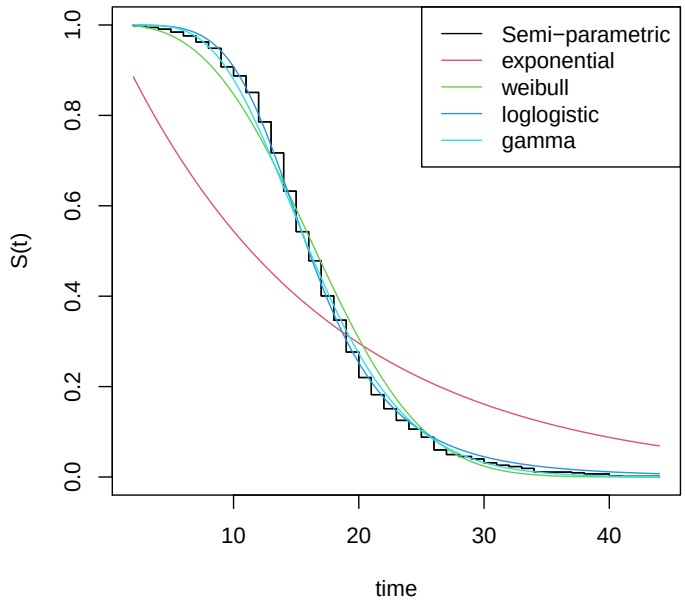
3.1 Examining Baseline Distribution

Althought the semi-parametric model is more flexible, and thus more robust to unusual baseline distributions, there are many reasons one may decide to use a parametric model instead. One reasons is that, as stated earlier, we are not aware of any general distributional theory regarding the baseline distribution, outside of the univariate case with case I interval censored data. Even in this case, the estimator is highly inefficient, observing convergence rates of $n^{1/3}$ instead of the more standard $n^{1/2}$. Because of this, making inference about values that directly require the baseline distribution, such as creating a confindence interval for the median for subjects with a given set of covariates, cannot be done with the semi-parametric model.

However, even if a parametric model is used for final inference, the semi-parametric model is still useful for assessing model fit. This is especially important for interval censored data, as we do not have the option of examining typical residuals or histograms as we would if the outcome was uncensored. **icenReg** has the function `diag_baseline` that plots several choices of parametric baseline distributions against the semi-parametric estimate. If the parametric distribution shows no systematic deviations from the semi-parametric fit, this implies the choice of parametric family may do a reason job of describing the underlying distribution. If there are clear deviations, this model should not be trusted.

To use `diag_baseline`, you must provide either a fitted model, or a formula, data and model. You then select the parametric families that you would like plotted against the non-parametric estimate (default is to fit all available). As an example, suppose we wanted to examine the different parametric fits for the `IR_diabetes` dataset. This could be done with

```
> diag_baseline(cbind(left, right) ~ gender,
+     model = "po",
+     data = IR_diabetes,
+     dists = c("exponential", "weibull",
+               "loglogistic", "gamma"),
+     lgdLocation = "topright")
```



Alternatively, using the fits from earlier, we can just call

```
> diag_baseline(fit_po, lgdLocation = "topright",
+               dists = c("exponential", "weibull",
+                         "loglogistic", "gamma")
+               )
```

Visual diagnostics are always subjective, but in this case we definitively know that the exponential fit is not appropriate and we believe the gamma baseline is most appropriate for the proporitonal odds model (although there is not overwhelming evidence that it is best).

3.2 Examining Covariate Effect

Although semi-parametric models do not make assumptions about the parametric family of the baseline distribution, both fully-parametric and semi-parametric models make assumptions about the form of the covariate effect, akin to the link function in generalized linear models.

A rule of thumb for identifying gross violations of proportional hazards is to check if the Kaplan Meier curves cross; if they do, and this cross appears not purely by chance, the proportional hazards assumption seems inappropriate.

This can naturally extend to the case of interval censored data by replacing the Kaplan Meier curves with the NPMLE. Also, this informal test can be generalized to the proportional odds model; the proportional odds assumption also implies that survival curves that differ only by a constant factor of the odds of survival should not cross.

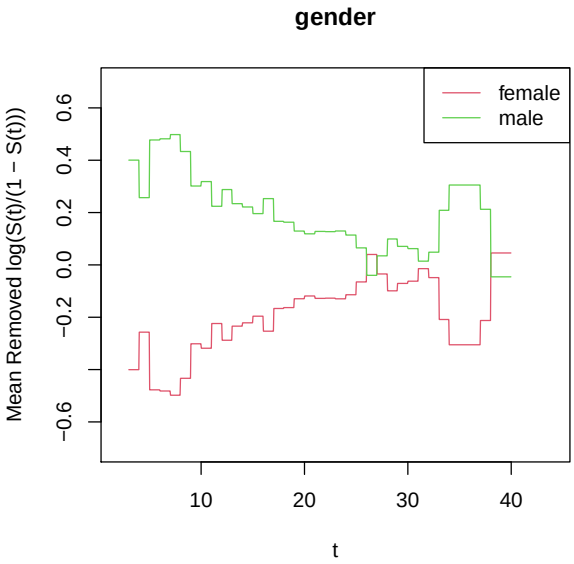
Another method of assessing involves transforming yoursurvival estimates such that if the assumptions are met, the difference in transformed survival will be constant. For the proportional hazards model, this is the complementary log-log tranformation (i.e. $\log(-\log(s))$). For the proportional odds model, this is the logit transformation (i.e. $\log(s/(1-s))$).

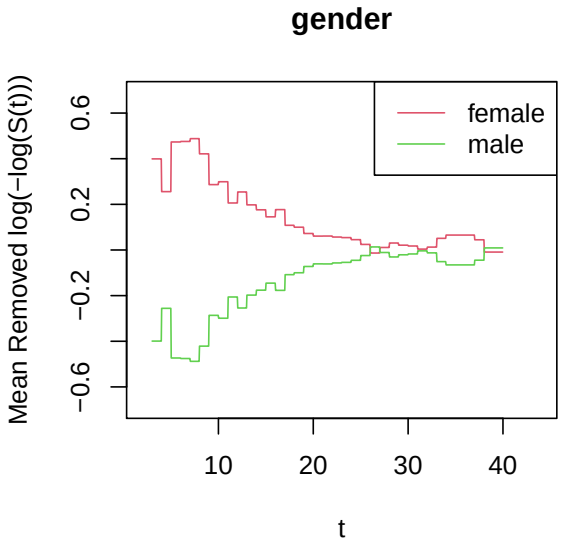
Plotting these functions can be done automatically in **icenReg** using the `diag_covar` function. The basic flow is that function takes in the fit, divides the data up on a covariate of interest. If it is categorical, it simply breaks up by category, if it is numeric, it attempts to find break point to evenly split up the data. Then, for each subset of the data, it fits the corresponding semi-parametric model and plots the transformation of the baseline distribution.

To demonstrate, suppose we wanted to assess whether the Cox-PH or proportional odds model was more appropriate for the `IR_diabetes`. This could be done by

```
> diag_covar(fit_po, lgdLocation = "topright",
+            main = "Checking Proportional Odds")
> diag_covar(fit_ph, lgdLocation = "topright",
+            main = "Checking Proportional Hazards")
```

We see that especially for gender, the porportional odds seems somewhat more appropriate (the difference between transformed values seems more constant). This agrees with the fact that the likelihood is approximately 2.5 greater for the proportional odds model than Cox-PH.





4 Appendix

4.1 Parallel Bootstrapping

Bootstrapping can be very computationally intensive. Fortunately, it is also embarrassingly parallel. As such, icenReg is written to work seamlessly with `doParallel`

```
> library(doParallel)
> myCluster <- makeCluster(4)  #uses 4 cores
```

```
> registerDoParallel(myCluster)
> fit <- ic_sp(cbind(left, right) ~ gender,
+            data = IR_diabetes, model = "po",
+            bs_samples = 50, useMCores = TRUE)
>stopCluster(myCluster)
```