Using icenReg for Customer Abandonment Analysis

Clifford Anderson-Bergman

The goal of this document is to show how **icenReg** can be used to analyze customer abandonment data.

```
library(icenReg, quietly = T)
set.seed(1)
```

In this situation, customers approach a service center and are given a ticket. When the service center is ready, they call the customer's ticket. However, some of the customers leave before their ticket is called. We don't observe the time when the customers leave, but we do record the service time and whether the customer has left, along with other variables that may affect both service time and customer patience.

We start by writing a function that simulates some customer data. In our simulation, each subject has a patience time (how long they will stay until they decide to leave) and a service time (how long they had to wait for service). We don't actually get to see any of the customers' patience time, but we see the service time (just named "time" in the dataset produced) and whether they decided to leave by the service time ("left"). If they left before the service time, we can assume that the patience time is in the interval (0, service time). If they have not left by the service time, we can assume that the patience time is in the interval (service time, ∞).

In addition, we have two covariates; "busy" (whether the center was busy when the customers approached) and "numServers" (number of servers working at a center). In our simulation, when its busy, service takes longer, but customers are slightly more patient. The number of servers does not affect a subject's patience, but does reduce service time.

We note that this is an overly simplistic/naive model. For example, number of servers working is independent of whether the center is busy. But this naive model should work fine for demonstrating how to use **icenReg** for this type of analysis.

```
simWaits <- function(n = 100,
                     busyEffectOnService =  log(3),
                     busyEffectOnPatience = log(1.5),
                     numServersEffectOnService = log(0.7) ){
  isBusy <- rbinom(n, size = 1, prob = 0.5)
  numServers <- rpois(n, lambda = 1) + 1
  patience_time_adj <- exp(isBusy * busyEffectOnPatience)
  patience_time <- rexp(n) * patience_time_adj * 2
  service_adj <- exp(isBusy * busyEffectOnService +
                       numServers * numServersEffectOnService)
  service_time <- rexp(n) * service_adj
  res <- data.frame(time = service_time,
                    left = service_time > patience_time,
                    numServers = numServers,
                    busy = isBusy)
  return(res)
}

exampleData <- simWaits(n = 4000)

head(exampleData)
```

```
##        time  left numServers busy
```

```
## 1 0.2491973 FALSE          1    0
## 2 0.7472399 FALSE          2    0
## 3 2.6546208  TRUE          1    1
## 4 0.3429795 FALSE          2    1
## 5 0.2110561 FALSE          4    0
## 6 1.7936230 FALSE          2    1
```

Next, we fit the survival model for subject's patience. The dataset is in current status format, so we can use the `cs2ic` function to convert this into current status format.

We have chosen to use a Bayesian AFT model with a Weibull baseline distribution, but this could work just as well for any of the other regression models in **icenReg**.

```
patience_fit <- ic_bayes(cs2ic(time, left) ~ numServers + busy,
                         data = exampleData,
                         model = 'aft')
```

We now have a model of patience. To answer the question of abandoment, we need to model how long service will take.

The service time is not censored, so we are free to use any modeling tool we would like. However, there's nothing wrong with using interval-censored methods on uncensored data, so we will also model the serivce time using **icenReg**. If a time is observed exactly at time t, it can be represented as (t, t) in the interval censored format.

```
service_fit <- ic_bayes(cbind(time,time) ~ numServers + busy,
                        data = exampleData,
                        model = 'aft')
```

Now suppose we want to know the probability of abandonment under various scenarios. We can make a new dataset with the covariate settings we are interested in.

```
newdata <- data.frame(numServers = c(1, 1, 5),
                      busy = c(0, 1, 1))
rownames(newdata) <- paste("Scenario", 1:3)
newdata
```

```
##            numServers busy
## Scenario 1          1    0
## Scenario 2          1    1
## Scenario 3          5    1
```

We then take 1000 samples of both the patience time and service time under each of these scenarios. Then by simply taking the average number of times that patience time was less than service time, we have an estimate for the probability of abandonment.

```
patience_samples <- ic_sample(patience_fit, newdata, samples = 1000)
service_samples <- ic_sample(service_fit, newdata, samples = 1000)

rowMeans(patience_samples < service_samples)
```

```
## Scenario 1 Scenario 2 Scenario 3
##      0.251      0.397      0.162
```

The default behavior for `ic_sample` is to first take a sample from the posterior of the distribution of model parameters and then take a sample of the response variables conditional on the parameters. As such, this produces the posterior probability of abandonment. Alternatively, we can just use the MAP estimates for the model parameters for every sample by setting `sampleType = 'fixedParSample'`.

```
patience_samples <- ic_sample(patience_fit, newdata, samples = 1000,
                              sampleType = 'fixedParSample' )
service_samples <- ic_sample(service_fit, newdata, samples = 1000,
                             sampleType = 'fixedParSample')

rowMeans(patience_samples < service_samples)
```

```
## Scenario 1 Scenario 2 Scenario 3
##      0.253      0.399      0.172
```