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.

Simulating the data

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 it is 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),
                     probTalk = 0.0){
  # Simulating covariates
  isBusy <- rbinom(n, size = 1, prob = 0.5)
  numServers <- rpois(n, lambda = 1) + 1

  # Simulating patience and service time
  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
  left <- service_time > patience_time

  # Simulating biasing effect; discussed later
  if(probTalk > 0){
    left_ind <- which(left)
    talked <- rbinom(length(left_ind), size = 1, prob = probTalk)
```

```
    talked_ind <- left_ind[talked == 1]
    service_time[talked_ind] <- patience_time[talked_ind]
  }
  res <- data.frame(time = service_time,
                    left = left,
                    numServers = numServers,
                    busy = isBusy)
  return(res)
}

# True values of parameters used in simulation model
true_patience_params <- c(0,log(2),0, log(1.5))
true_service_params  <- c(0,0,log(0.7), log(3))

# Data we will use
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
```

Fitting models

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')
```

Because this is a simulation, we can look and see how our estimates compared to the true parameters used to simulate the data.

```
coef_res <- list()
patience_results <- rbind(coef(patience_fit),
```

```
                          sqrt(diag(vcov(patience_fit))),
                          true_patience_params)
rownames(patience_results) <- c('Estimated Parameters',
                                'Posterior SD',
                                'True Parameters')

service_results  <- rbind(coef(service_fit),
                    sqrt(diag(vcov(service_fit))),
                    true_service_params)
rownames(service_results) <- c('Estimated Parameters',
                                'Posterior SD',
                                'True Parameters')

coef_res[['Patience Model']] <- patience_results
coef_res[['Service Model']] <- service_results

print(coef_res, digits = 3)
```

```
## $`Patience Model`
##                       log_shape log_scale numServers   busy
## Estimated Parameters   -0.0430    0.8243    -0.0409 0.3190
## Posterior SD            0.0407    0.0884     0.0377 0.0726
## True Parameters         0.0000    0.6931     0.0000 0.4055
##
## $`Service Model`
##                       log_shape log_scale numServers   busy
## Estimated Parameters   0.00764   -0.0369    -0.3472 1.1290
## Posterior SD           0.01255    0.0389     0.0156 0.0313
## True Parameters        0.00000    0.0000    -0.3567 1.0986
```

It's worth noting that the posterior standard deviation is considerably smaller for the service model compared with the patience model. This is natural; we observed the service times exactly, where as the patience time we only know up to a potentially very large interval. So it makes sense that our data is more informative about the distribution of service time than about the patience time.

Estimating the probability of abandoment

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)
res <- list(`Estimated Probabilities of Abandonment` =
              rowMeans(patience_samples < service_samples) )
res
```

```
## $`Estimated Probabilities of Abandonment`
## 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')

res <- list(`Estimated Probabilities of Abandonment` =
              rowMeans(patience_samples < service_samples) )
res
```

```
## $`Estimated Probabilities of Abandonment`
## Scenario 1 Scenario 2 Scenario 3
##      0.253      0.399      0.172
```

Potential confounder: non-independence

Our model makes the assumption that service time and patience time are *conditionally* independent. This is one of the very common assumptions of interval censored models; that the process that generates the inspection times is conditionally independent of the event times. Note that the service time and patience time certainly are not marginally indepedent; when it is busy, both the service time and patience times are larger. But since "busy" is captured in the model, this does not affect inference.

But let's consider a scenario where this could happen. Suppose that some of our customers leave and tell the servers to cancel their ticket. For the sake of ruining our data, let's assume the server then marks this as the service time (the ticket is removed) and does not tell us what happened. In this case, there is a dependency between the service time and patience time which is not captured in the data itself. This will bias our results. The extent of the bias is depedent on how inapproriate our model is.

To demonstrate, we will simulate two datasets. In the first, we will simulate that of the subjects who left, 10% talked with servers (i.e. service time == patience time, but in the data we see, these are right censored). We also simulate a dataset where 75% talked with servers.

```
light_bias_data <- simWaits(n = 4000,
                            probTalk = 0.1)
heavy_bias_data <- simWaits(n = 4000,
                            probTalk = 0.75)

light_bias_patience_fit <- ic_bayes(cs2ic(time, left) ~ numServers + busy,
                          data = light_bias_data,
                          model = 'aft')
heavy_bias_patience_fit <- ic_bayes(cs2ic(time, left) ~ numServers + busy,
                          data = heavy_bias_data,
```

```
                        model = 'aft')


coef_res <- list()
light_bias_results <- rbind(coef(light_bias_patience_fit),
                            sqrt(diag(vcov(light_bias_patience_fit))),
                            true_patience_params)
rownames(light_bias_results) <- c('Estimated Parameters',
                                  'Posterior SD',
                                  'True Parameters')
heavy_bias_results <- rbind(coef(heavy_bias_patience_fit),
                            sqrt(diag(vcov(heavy_bias_patience_fit))),
                            true_patience_params)
rownames(heavy_bias_results) <- c('Estimated Parameters',
                                  'Posterior SD',
                                  'True Parameters')

res <- list(`Light Dependence` = light_bias_results,
            `Heavy Dependence` = heavy_bias_results)
print(res, digits = 3)
```

```
## $`Light Dependence`
##                       log_shape log_scale numServers   busy
## Estimated Parameters   -0.2180     0.740     0.1312 0.2436
## Posterior SD            0.0454     0.103     0.0492 0.0856
## True Parameters         0.0000     0.693     0.0000 0.4055
##
## $`Heavy Dependence`
##                       log_shape log_scale numServers   busy
## Estimated Parameters    -2.100     7.044      2.068 -4.017
## Posterior SD             0.215     1.921      0.585  1.278
## True Parameters          0.000     0.693      0.000  0.405
```

Sure enough, we see that if we have a depedence between the censoring mechanism (i.e. service) and the event time (i.e. customer's patience limit) that is *not* captured in the model, we can have bias in our estimated parameters.