Notes on Discrete Choice with Capacity Constraints

Nick Mader, Chapin Hall at the University of Chicago (nmader@chapinhall.org)

April 6, 2017

Statement of the Problem

In policy problems around takeup of human services, capacity constraints of service providers may lead to underestimation of the full demand for services, if households do not take up services if constrained out of enrollment. In this brief, I consider the example of family enrollment of children in Head Start ("HS") programming.

Let i generically index children to be enrolled in programming, with a total number of N children. Children and families have traits x_i, and their constrained choice to enroll in HS is

$$y_i = 1 \iff y_i^* \geq 0 y_i^* = x_i' \beta + \epsilon_i$$

Where the parameter vector β represents weights translating observed characteristics into latent propensity to seek enrollment in services.

However, we now impose a capacity constraint where

$$\sum_{j=1}^{N} y_i \leq \bar{n}$$

. Thus, an acceptance rule must be imposed, where by families with y_i^* are allowed to enroll. This rule for assigning probability of enrollment for a given family r_i is dependent on other families with $y_i^* \geq 0$:

$$r_i(y_i^* \| \{\{y_j^*\} \| y_j^* \geq 0 \forall j = 1, \dots, N\}) = f(y_{j \neq i}^*), f \in (0,1) \quad if \quad y_i^* \geq 0 0 \quad if \quad y_i^* < 0$$

where the full

Single Provider with Random Acceptance

We develop a sample economy with $N = 100$ agents, a single provider, provider capacity of $\bar{n} = 25$, and assigment rule of

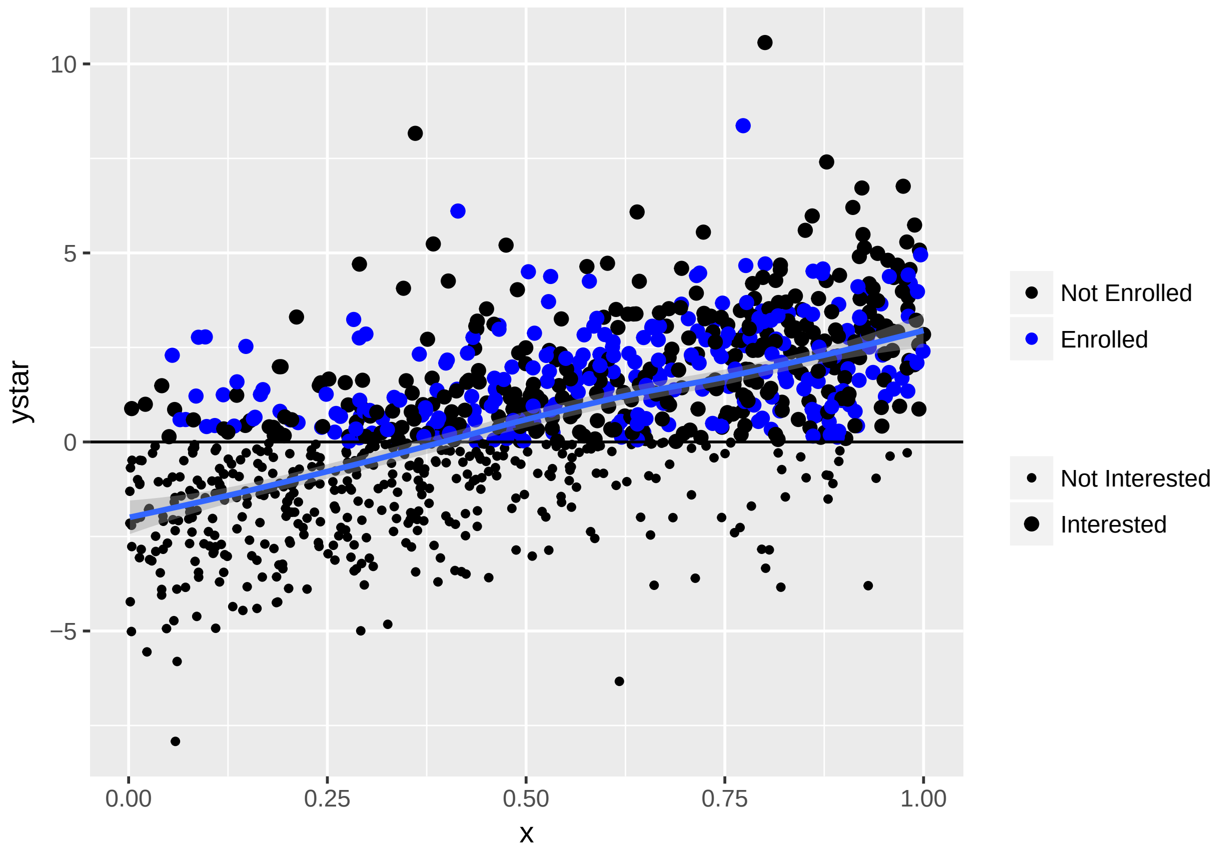
$$r_i(\mathbf{y}_j^*, \forall \{j | y_j^* \geq 0\}) = \begin{matrix} 0 & if \quad y_i^* < 0 \\ \bar{n}/\tilde{n},\ \tilde{n} = max(\Sigma_j \{y_j^* \geq 0\},\ \bar{n}) & if \quad y_i^* \geq 0 \end{matrix}$$

to demonstate the bias of conventional methods.

```r
set.seed(60607)
N <- 1000
nbar <- 250
b <- 5
a <- -2

RunLottery <- function(ystars, nbar){
  ystars_ge0 <- 1*(ystars > 0)
  nInterested <- sum(ystars_ge0)
  prob <- nbar / nInterested

  draw <- runif(length(ystars_ge0))
  draw <- ifelse(ystars_ge0 == 0, 1, draw)
  enr <- ifelse(ystars_ge0 == 0, 0, 1*(draw <= sort(draw)[nbar]))
  return(enr)
}

MakeData <- function(){

  # Draw basic features
  x <- runif(N)
  e <- rlogis(N)
  ystar = a + x*b + e
  ystar_ge0 = 1*(ystar >= 0)

  # Generate data set
  dt <-
    data.table(x = x,
               e = e,
               ystar = ystar,
               ystar_ge0 = ystar_ge0)

  # Add capacity contraint
  dt$y <- RunLottery(ystars = dt$ystar, nbar = nbar)


  return(dt)
}

#summary(dt$ystar_ge0)
#sum(dt$y)

dt <- MakeData()

dt[, summary(ystar_ge0)]
```

```
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##   0.000   0.000   1.000   0.596   1.000   1.000
```

```r
ggplot(dt, aes(x = x, y = ystar)) +
  geom_point(aes(size = factor(ystar_ge0), colour = factor(y))) +
  geom_hline(yintercept = 0) +
  geom_smooth(method = "loess") +
  scale_size_manual(name = "", values = c(1, 2), labels = c("Not Interested", "Interested")) +
  scale_colour_manual(name = "", values = c("black", "blue"), labels = c("Not Enrolled", "Enrolled"))
```



```r
simpleReg <- glm(y ~ x, data = dt, family = "binomial")
tidy(simpleReg)
```

```
##          term  estimate std.error  statistic      p.value
## 1 (Intercept) -2.190171 0.1787822 -12.250495 1.669743e-34
## 2           x  1.999748 0.2779317   7.195107 6.241168e-13
```

The proper likelihood function is

$$\mathcal{L} = \prod_i^N \Big[pr(y_i^* < 0) + pr(y_i^* > 0)\Big(1 - \bar{n}/\tilde{n}\Big)\Big]^{y_i=0} \Big[pr(y_i^* > 0)\Big(\bar{n}/\tilde{n}\Big)\Big]^{y_i=1}$$

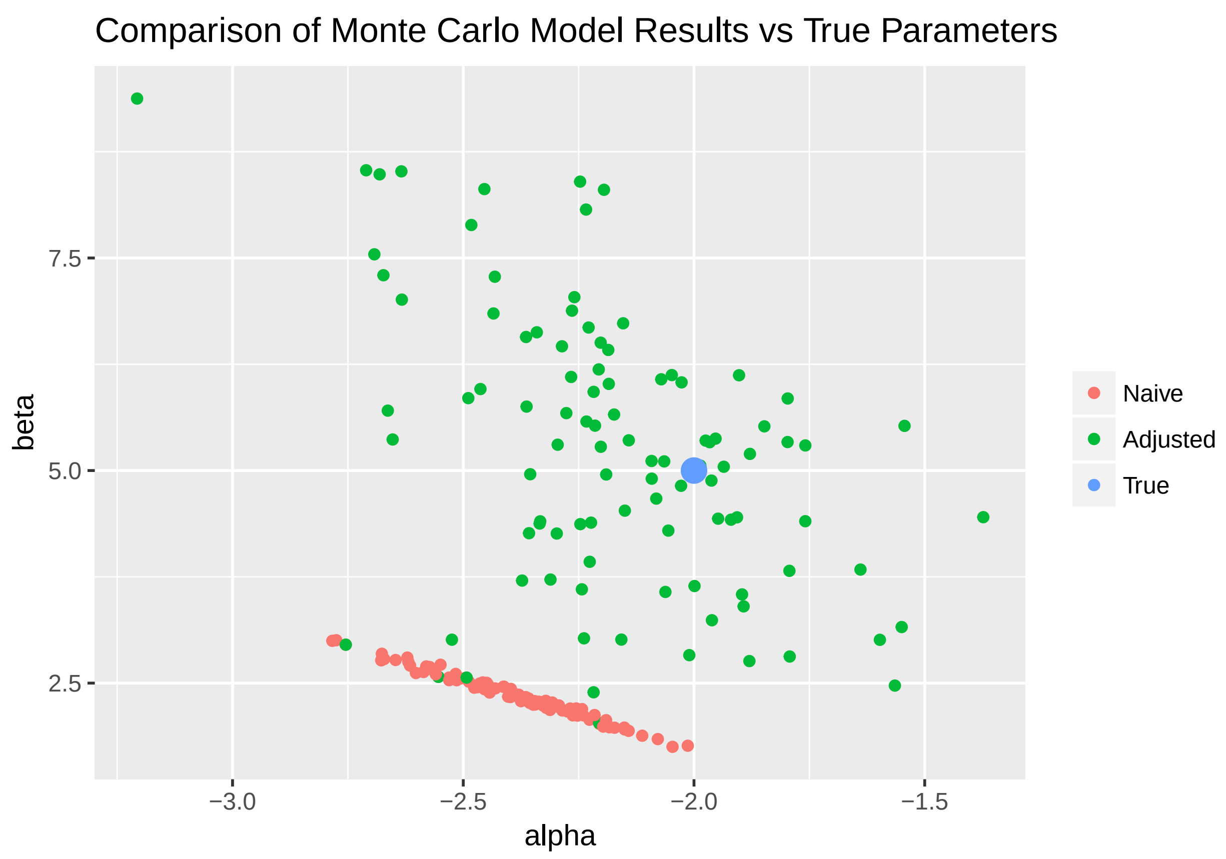
$$\tilde{n} = \sum_i^N (y_i^* > 0)$$

```r
fullLike <- function(beta, dt){
  xb <- beta[1] + dt$x*beta[2]
  pr_ysGe0 <- plogis(xb)
  r <- min(1, nbar/sum(pr_ysGe0)) # probability of being accepted

  pr_yLt0 <- (1-pr_ysGe0) + pr_ysGe0*(1-r)
  pr_yGe0 <- pr_ysGe0*r

  ll <- (1-dt$y)*log(pr_yLt0) + dt$y*log(pr_yGe0)
  return(-sum(ll))
}
```

```r
# Run a range of data draw scenarios to run estimation with

runs <- 100
m <- matrix(rep(0, 4*runs), ncol = 2)
mypar <- data.frame(alpha = m[, 1], beta = m[, 2], stringsAsFactors = FALSE)
for (i in 1:100){
  if (i %% 10 == 0) print(paste("Working on run", i))
  dt <- MakeData()

  # Run naive
  logistic <- glm(y ~ x, data = dt, family = "binomial")
  mypar[2*i-1, ] <- c(logistic$coefficients)

  # Run capacity-aware estimation
  opt <- optim(c(0, 0), fullLike, dt = dt)
  mypar[2*i,] <- c(opt$par)
}
```

```
## [1] "Working on run 10"
## [1] "Working on run 20"
## [1] "Working on run 30"
## [1] "Working on run 40"
## [1] "Working on run 50"
## [1] "Working on run 60"
## [1] "Working on run 70"
## [1] "Working on run 80"
## [1] "Working on run 90"
## [1] "Working on run 100"
```

```r
mypar$spec <- c("naive", "adj")

true <- data.frame(alpha = a, beta = b, spec = "true", stringsAsFactors = FALSE)
mypar <- rbind(mypar, true) %>%
  within(spec <- factor(spec, levels = c("naive", "adj", "true")))

# Display results

# /!\ Look for how to change the order of
ggplot(data = mypar, aes(x = alpha, y = beta, color = factor(spec), size = factor(spec))) +
  geom_point() +
  scale_size_manual(values = c(1.5, 1.5, 4), guide = FALSE) +
  scale_color_discrete(name = "", labels = c("Naive", "Adjusted", "True")) +
  labs(x = "alpha", y = "beta") +
  ggtitle("Comparison of Monte Carlo Model Results vs True Parameters")
```



Extension to Non-Random Admission

Extension to Multiple Service Providers