choice-frequency-simulation

```
library(tidyverse)
library(mvtnorm)
library(mnormt)
```

Introduction

This paper does a simulation study to learn about the simultaneous equation model as implemented in Pouri and Bhat (2003). The modeling framework is actually a sample selection (Heckman type model) where the selection equation is binary (as usual) but the outcome equation features a discrete (ordered) choice (instead of a continuous one).

You may also want to consult `sample-selection.Rmd` for more background about the modeling framework (there a Tobit-2-model).

We follow their notation and the model equations read:

$$
\begin{aligned}
t_i^* &= \gamma'\mathbf{X_i} + \varepsilon_i, \quad t_i = 1 \text{ if } t_i^* > 0 \text{ and } t_i = 0 \text{ otherwise} && (1)\\
N_i^* &= \alpha'\mathbf{Z_i} + \eta_i, \quad N_i = j \text{ if } a_{j-1} < N_i^* \leq a_j, && (2)\\
&\qquad j = 1, 2, \ldots, J, \quad N_i \text{ observed only if } t_i^* > 0
\end{aligned}
$$

The error terms are assumed to follow a bivariate normal distribution. The probability that individual i telecommutes ($t_i = 1$) and does so for j days is:

$$
\begin{aligned}
P(t_i = 1, N_i = j) = &\ \Phi_2(a_j - \alpha'\mathbf{Z_i}; \gamma'\mathbf{X_i}; -\rho) && (3)\\
&- \Phi_2(a_{j-1} - \alpha'\mathbf{Z_i}; \gamma'\mathbf{X_i}; -\rho)
\end{aligned}
$$

Let's define a set of dummy variables M_{ij}:

$$
M_{ij} = \begin{cases} 1 & \text{if } N_i = j (\text{i.e.}, a_{j-1} < N_i^* \leq a_j) \\ 0 & \text{otherwise} \end{cases} \qquad (4)
$$

This yields the following maximum likelihood (not so nice notation...):

$$
\begin{aligned}
L = \prod_{i=1}^{I} \Bigg\{ [1 - \Phi(\gamma'\mathbf{X_i})]^{1-t_i} \prod_{j=1}^{J} [\Phi_2(a_j - \alpha'\mathbf{Z_i}; \gamma'\mathbf{X_i}; -\rho) && (5)\\
- \Phi_2(a_{j-1} - \alpha'\mathbf{Z_i}; \gamma'\mathbf{X_i}; -\rho)]^{M_{ij}} \Bigg\}^{t_i}
\end{aligned}
$$

See also De Luca and Perotti (2011) and Toomet and Henningsen (2008)

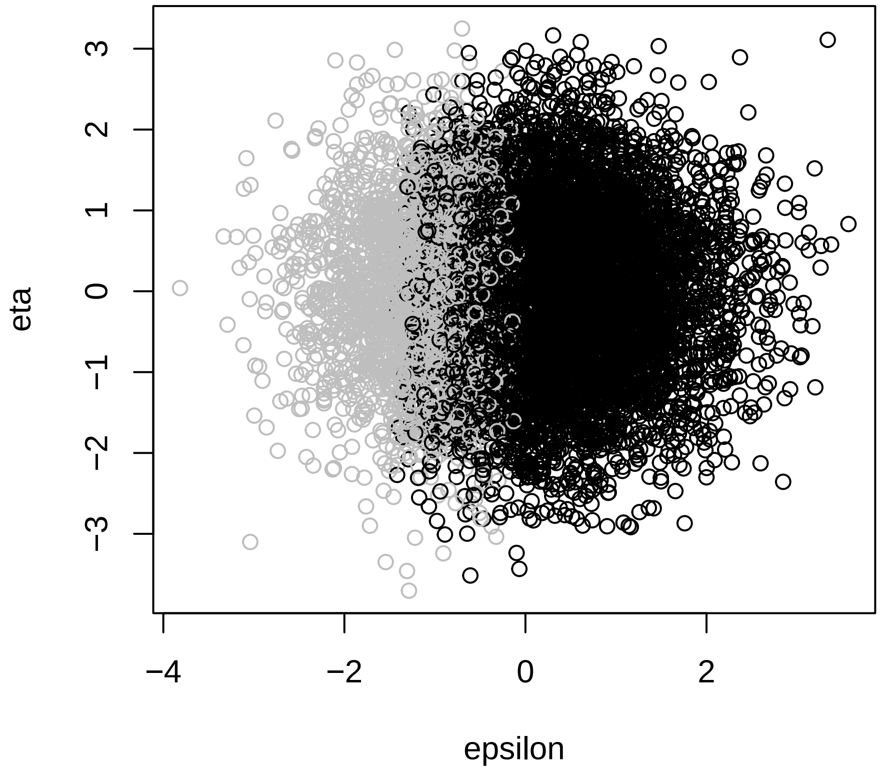
Simulation

```
n <- 10e3
simulate_data <- function(rho, n) {
  out <- list()
  # Parameters
  n <- n
  gamma <- 1.5
  alpha <- 2
  rho <- rho
  a1 <- 0.5
  a2 <- 1.5

  ground_truth <- c(gamma = gamma, alpha = alpha, rho = rho, a1 = a1, a2 = a2)

  # Errors
  set.seed(0)
  errors <- rmvnorm(n, c(0, 0), sigma = matrix(c(1, rho, rho, 1), 2, 2))
  epsilon <- errors[, 1]
  eta <- errors[, 2]

  # Data generating process
  X <- runif(n)
  t_star <- gamma * X + epsilon
  t <- t_star > 0
  Z <- runif(n)
  N_star <- alpha * Z + eta
  N <- cut(N_star, breaks = c(-Inf, a1, a2, Inf))
  levels(N) <- c(1, 2, 3)
  N <- as.numeric(as.character(N))
  N <- N * t

  # Model frame
  dat <- data.frame(
    X = X,
    Z = Z,
    t = as.numeric(t),
    N = N
  )

  out$ground_truth <- ground_truth
  out$errors <- errors
  out$data <- dat

  return(out)
}
```

No error correlation rho=0

```
sim_dat <- simulate_data(rho = 0, n = n)
sim_dat$ground_truth
#> gamma alpha   rho    a1    a2
#>   1.5   2.0   0.0   0.5   1.5
```

```
dat <- sim_dat$data
plot(sim_dat$errors, col = ifelse(dat$t, "black", "grey"),
     xlab = "epsilon",
     ylab = "eta")
```




```
table(dat$t)
#>
#>    0    1
#> 2410 7590
table(dat$N)
#>
#>    0    1    2    3
#> 2410 2574 2463 2553
head(dat)
#>           X         Z t N
#> 1 0.2984493 0.6230550 1 2
#> 2 0.5930309 0.5779149 1 3
#> 3 0.5528832 0.7006375 1 1
#> 4 0.5967040 0.3372637 0 0
#> 5 0.3926307 0.1190206 1 3
#> 6 0.8981738 0.8776399 1 2
```

Estimate binary probit

```
fit <- stats::glm(t ~ X, data = dat, family = binomial(link = "probit"))
summary(fit)
#>
```

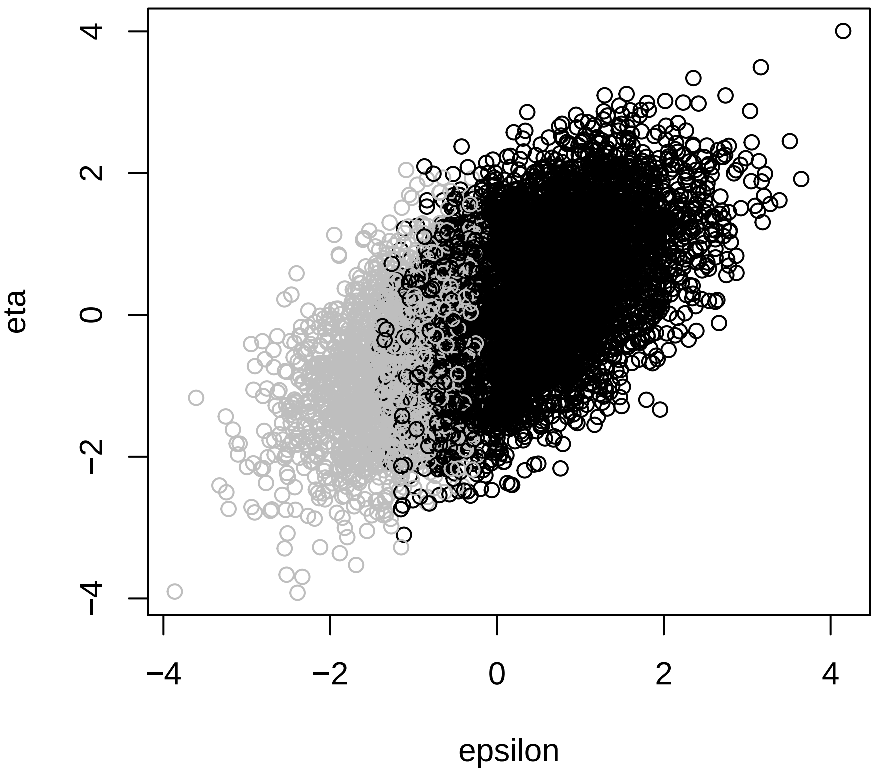
```
#> Call:
#> stats::glm(formula = t ~ X, family = binomial(link = "probit"),
#>     data = dat)
#>
#> Coefficients:
#>             Estimate Std. Error z value Pr(>|z|)
#> (Intercept) -0.02141    0.02679  -0.799    0.424
#> X            1.57840    0.05213  30.281   <2e-16 ***
#> ---
#> Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
#>
#> (Dispersion parameter for binomial family taken to be 1)
#>
#>     Null deviance: 11045  on 9999  degrees of freedom
#> Residual deviance: 10062  on 9998  degrees of freedom
#> AIC: 10066
#>
#> Number of Fisher Scoring iterations: 4
```

Estimate ordered probit

```
dat_ <- dat[dat$t == 1, ]
dat_$N <- factor(dat_$N)
fit <- MASS::polr(N ~ Z, data = dat_, method = "probit")
summary(fit)
#>
#> Re-fitting to get Hessian
#> Call:
#> MASS::polr(formula = N ~ Z, data = dat_, method = "probit")
#>
#> Coefficients:
#>   Value Std. Error t value
#> Z 2.041    0.04822   42.33
#>
#> Intercepts:
#>     Value   Std. Error t value
#> 1|2  0.5304  0.0272    19.4894
#> 2|3  1.5188  0.0305    49.8178
#>
#> Residual Deviance: 14789.89
#> AIC: 14795.89
```

Error correlation rho > 0

```
sim_dat_ <- simulate_data(rho = 0.6, n = n)
sim_dat_$ground_truth
#> gamma alpha   rho    a1    a2
#>   1.5   2.0   0.6   0.5   1.5
dat <- sim_dat_$data
plot(sim_dat_$errors, col = ifelse(dat$t, "black", "grey"),
     xlab = "epsilon",
     ylab = "eta")
```



Estimate binary probit

```
fit <- stats::glm(t ~ X, data = dat, family = binomial(link = "probit"))
summary(fit)
#>
#> Call:
#> stats::glm(formula = t ~ X, family = binomial(link = "probit"),
#>     data = dat)
#>
#> Coefficients:
#>             Estimate Std. Error z value Pr(>|z|)
#> (Intercept) 0.003342   0.026739   0.125    0.901
#> X           1.508402   0.051644  29.208   <2e-16 ***
#> ---
#> Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
#>
#> (Dispersion parameter for binomial family taken to be 1)
#>
#>     Null deviance: 11072  on 9999  degrees of freedom
#> Residual deviance: 10163  on 9998  degrees of freedom
#> AIC: 10167
#>
#> Number of Fisher Scoring iterations: 4
```

Estimate ordered probit

```
dat_ <- dat[dat$t == 1, ]
dat_$N <- factor(dat_$N)
```

```
fit <- MASS::polr(N ~ Z, data = dat_, method = "probit")
summary(fit)
#>
#> Re-fitting to get Hessian
#> Call:
#> MASS::polr(formula = N ~ Z, data = dat_, method = "probit")
#>
#> Coefficients:
#>   Value Std. Error t value
#> Z 2.176    0.04918   44.24
#>
#> Intercepts:
#>     Value   Std. Error t value
#> 1|2  0.3131  0.0270    11.6061
#> 2|3  1.3815  0.0301    45.9385
#>
#> Residual Deviance: 14324.62
#> AIC: 14330.62
```

The estimates are **upward** biased!

Own implementation binary probit

Again, back to no error correlation.

```
sim_dat$ground_truth
#> gamma alpha   rho    a1    a2
#>   1.5   2.0   0.0   0.5   1.5
dat <- sim_dat$data
dat_ <- dat[dat$t == 1, ]
n_ <- nrow(dat_)

loglik <- function(param) {
  gamma <- param["gamma"]
  gammaX <- gamma * dat$X
  p1 <- pnorm(gammaX)
  p0 <- 1 - p1
  ll <- sum((1 - dat$t) * log(p0) + dat$t * log(p1))
  # cat(ll, "\n")
  ll
}

m <- maxLik::maxLik(loglik, start = c(gamma = 0))
summary(m)
#> --------------------------------------------
#> Maximum Likelihood estimation
#> Newton-Raphson maximisation, 4 iterations
#> Return code 8: successive function values within relative tolerance limit (reltol)
#> Log-Likelihood: -5031.553
#> 1  free parameters
#> Estimates:
#>       Estimate Std. error t value Pr(> t)
```

```
#> gamma  1.54312    0.02791    55.3  <2e-16 ***
#> ---
#> Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
#> --------------------------------------------
```

Own implementation ordered probit

```
loglik <- function(param) {
  alpha <- param["alpha"]
  a1 <- param["a1"]
  a2 <- param["a2"]
  alphaZ <- alpha * dat_$Z
  # could be done outside once and for all...
  I1 <- as.numeric(dat_$N == 1)
  I2 <- as.numeric(dat_$N == 2)
  I3 <- as.numeric(dat_$N == 3)
  p1 <- pnorm(a1 - alphaZ)
  p2 <- pnorm(a2 - alphaZ) - pnorm(a1 - alphaZ)
  p3 <- 1 - pnorm(a2 - alphaZ)
  ll <- sum(I1 * log(p1), I2 * log(p2), I3 * log(p3))
  # cat(ll, "\n")
  ll
}

m <- maxLik::maxLik(loglik, start = c(alpha =01, a1 = -1, a2 = 1))
summary(m)
#> --------------------------------------------
#> Maximum Likelihood estimation
#> Newton-Raphson maximisation, 6 iterations
#> Return code 2: successive function values within tolerance limit (tol)
#> Log-Likelihood: -7394.945
#> 3  free parameters
#> Estimates:
#>       Estimate Std. error t value Pr(> t)
#> alpha  2.04147    0.04802   42.51  <2e-16 ***
#> a1     0.53042    0.02714   19.54  <2e-16 ***
#> a2     1.51882    0.03038   50.00  <2e-16 ***
#> ---
#> Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
#> --------------------------------------------
```

Selection model

Now back to the main purpose: To estimate the selection model with ordered response data. Here we actually implement the log-likelihood (i.e. log-transforming the equation elaborated in the intro).

```
sim_dat_$ground_truth
#> gamma alpha   rho    a1    a2
#>   1.5   2.0   0.6   0.5   1.5
dat <- sim_dat_$data

# Compute once and for all itters
```

```r
I1 <- as.numeric(dat$N == 1)
I2 <- as.numeric(dat$N == 2)
I3 <- as.numeric(dat$N == 3)

loglik <- function(param) {
  alpha <- param["alpha"]
  gamma <- param["gamma"]
  rho <- param["rho"]
  a1 <- param["a1"]
  a2 <- param["a2"]
  gammaX <- gamma * dat$X
  alphaZ <- alpha * dat$Z
  sigma <- matrix(c(1, -rho, -rho, 1), 2, 2)
  n_p_inf <- rep(Inf, n)
  pt0 <- 1 - pnorm(gammaX)
  a1 <- pmnorm(cbind(gammaX, a1 - alphaZ), varcov = sigma)
  a2 <- pmnorm(cbind(gammaX, a2 - alphaZ), varcov = sigma)
  pt1_j1 <- a1 # below
  pt1_j2 <- a2 - a1 # between
  pt1_j3 <- pmnorm(cbind(gammaX, n_p_inf), varcov = sigma) - a2 # above
  selection <- (1 - dat$t) * log(pt0)
  observation <- dat$t * (I1 * log(pt1_j1) + I2 * log(pt1_j2) + I3 * log(pt1_j3))
  ll <- sum(selection + observation)
  # cat(ll, "\n")
  ll
}

loglik(param = sim_dat_$ground_truth)
#> [1] -12199.38

m <- maxLik::maxLik(loglik, start = c(gamma = 1, alpha = 1, rho = 0, a1 = -1, a2 = 1))
#> Warning in log(pt1_j3): NaNs produced

#> Warning in log(pt1_j3): NaNs produced

#> Warning in log(pt1_j3): NaNs produced

#> Warning in log(pt1_j3): NaNs produced

#> Warning in log(pt1_j3): NaNs produced

#> Warning in log(pt1_j3): NaNs produced

#> Warning in log(pt1_j3): NaNs produced

#> Warning in log(pt1_j3): NaNs produced

#> Warning in log(pt1_j3): NaNs produced

#> Warning in log(pt1_j3): NaNs produced

#> Warning in log(pt1_j3): NaNs produced

#> Warning in log(pt1_j3): NaNs produced
```

```
#> Warning in log(pt1_j3): NaNs produced

#> Warning in log(pt1_j3): NaNs produced

#> Warning in log(pt1_j3): NaNs produced

#> Warning in log(pt1_j3): NaNs produced

#> Warning in log(pt1_j3): NaNs produced

#> Warning in log(pt1_j3): NaNs produced

#> Warning in log(pt1_j3): NaNs produced

#> Warning in log(pt1_j3): NaNs produced

#> Warning in log(pt1_j3): NaNs produced

#> Warning in log(pt1_j3): NaNs produced

#> Warning in log(pt1_j3): NaNs produced

#> Warning in log(pt1_j3): NaNs produced

#> Warning in log(pt1_j3): NaNs produced

#> Warning in log(pt1_j3): NaNs produced

#> Warning in log(pt1_j3): NaNs produced

#> Warning in log(pt1_j3): NaNs produced
#> Warning in log(pt1_j2): NaNs produced
#> Warning in log(pt1_j3): NaNs produced
#> Warning in log(pt1_j2): NaNs produced
#> Warning in log(pt1_j3): NaNs produced
#> Warning in log(pt1_j2): NaNs produced
#> Warning in log(pt1_j3): NaNs produced
#> Warning in log(pt1_j2): NaNs produced
#> Warning in log(pt1_j3): NaNs produced
#> Warning in log(pt1_j2): NaNs produced
#> Warning in log(pt1_j3): NaNs produced
#> Warning in log(pt1_j2): NaNs produced

#> Warning in log(pt1_j2): NaNs produced
summary(m)
#> --------------------------------------------
#> Maximum Likelihood estimation
#> Newton-Raphson maximisation, 8 iterations
#> Return code 8: successive function values within relative tolerance limit (reltol)
#> Log-Likelihood: -12198.98
#> 5  free parameters
#> Estimates:
#>       Estimate Std. error t value Pr(> t)
```

```
#> gamma  1.51426    0.02759   54.89  <2e-16 ***
#> alpha  2.01357    0.05368   37.51  <2e-16 ***
#> rho    0.59133    0.04613   12.82  <2e-16 ***
#> a1     0.50522    0.02820   17.92  <2e-16 ***
#> a2     1.49743    0.02833   52.87  <2e-16 ***
#> ---
#> Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
#> --------------------------------------------
```

References

De Luca, Giuseppe, and Valeria Perotti. 2011. "Estimation of Ordered Response Models with Sample Selection." *The Stata Journal* 11 (2): 213–39.

Pouri, Yasasvi D, and Chandra R Bhat. 2003. "On Modeling Choice and Frequency of Home-Based Telecommuting." *Transportation Research Record* 1858 (1): 55–60.

Toomet, Ott, and Arne Henningsen. 2008. "Sample Selection Models in r: Package sampleSelection." *Journal of Statistical Software* 27: 1–23.