# Homework 3: Multinomial Logit Models

*Peng Peng*

*3/6/2019*

## Exercise 1: Data Description

```
# Average and dispersion of the product characteristics
#-----------------------------------------------------------------------

# A list of functions to apply across all columns
x = data[, 3:12]
f = function(x){
  list(mean(x), sd(x))
}

# Transpose the result matrix and add product characteristics
pr_char = data.frame(t(sapply(x, f))) %>%
    rename(mean = X1, sd = X2) %>%
    cbind(names(data[, 3:12])) %>%
    print()
```

```
##               mean         sd names(data[, 3:12])
## PPk_Stk  0.5184362  0.1505174             PPk_Stk
## PBB_Stk  0.5432103  0.1203319             PBB_Stk
## PFl_Stk    1.01502 0.04289519             PFl_Stk
## PHse_Stk 0.4371477  0.1188312            PHse_Stk
## PGen_Stk 0.3452819 0.03516605            PGen_Stk
## PImp_Stk 0.7807785  0.1146461            PImp_Stk
## PSS_Tub  0.8250895 0.06121159             PSS_Tub
## PPk_Tub   1.077409 0.02972613             PPk_Tub
## PFl_Tub   1.189376 0.01405451             PFl_Tub
## PHse_Tub 0.5686734   0.072455            PHse_Tub
```

```
# Market share of each product-----------------------------------------------------

# Match names to choices
product = names(data[, 3:12])

setnames(data, old = product, new = as.character(c(1:10)))

# Calculate market share of brands for Stk
share_stk = data %>%
  select(2:8) %>%
  filter(choice < 7) %>%
  group_by(choice) %>%
  summarise (n = n()) %>%
  mutate(share = n / sum(n)) %>%
  mutate(product = product[3:8]) %>%
  select(product, share) %>%
  print()
```

```
## # A tibble: 6 x 2
##    product      share
##      <chr>      <dbl>
## 1  PFl_Stk 0.47859079
## 2 PHse_Stk 0.18943089
## 3 PGen_Stk 0.06585366
## 4 PImp_Stk 0.16070461
## 5  PSS_Tub 0.08536585
## 6  PPk_Tub 0.02005420
```

```
# Calculate market share of brands for Tub
share_tub = data %>%
  select(2, 9:12) %>%
  filter(choice > 6) %>%
  group_by(choice) %>%
  summarise (n = n()) %>%
  mutate(share = n / sum(n)) %>%
  mutate(product = c("PSS_Tub", "PPk_Tub", "PFl_Tub", "PHse_Tub")) %>%
  select(product, share) %>%
  print()
```

```
## # A tibble: 4 x 2
##    product      share
##      <chr>      <dbl>
## 1  PSS_Tub 0.40897436
## 2  PPk_Tub 0.26025641
## 3  PFl_Tub 0.28846154
## 4 PHse_Tub 0.04230769
```

```
# Map observed attributes to choice----------------------------------------------

t1 = table(data$choice, data$Income)
t1 = cbind(as.matrix(product), t1)

t2 = table(data$choice, data$Fam_Size)
t2 = cbind(as.matrix(product), t2)

t3 = table(data$choice, data$Fs3_4)
t3 = cbind(as.matrix(product), t3)

t4 = table(data$choice, data$Fs5.)
t4 = cbind(as.matrix(product), t4)

t5 = table(data$choice, data$college)
t5 = cbind(as.matrix(product), t5)

t6 = table(data$choice, data$whtcollar)
t6 = cbind(as.matrix(product), t6)
```

## Exercise 2: First Model

A conditional logit model could be used to model the effect of price on demand. The probability of household $i$ choosing product $j$ can be written as:

$$Pr_{ij} = \frac{exp(X_{ij}\beta)}{\sum_{j=1}^{m} exp(X_{ij}\beta)},$$

where $x_ij$ is the price of the product facing each household.

For identification purpose, we normalize $\alpha_1$ to 0. We divide the probability by $\frac{exp(X_{i1}\beta+\alpha_1)}{exp(X_{i1}\beta+\alpha_1)}$ we get:

$$Pr_{ij} = \frac{exp((X_{ij} - X_{i1})\beta + (\alpha_j - \alpha_1))}{\sum_{1}^{m} exp((X_{ik} - X_{i1})\beta + (\alpha_k - \alpha_1))}.$$

$$L = \prod_{i=1}^{n}\prod_{j=1}^{m} Pr_{ij}\mathbb{1}[j = j*]$$

where $j*$ is the actual choice made by $i$. Log-likelihood is thus:

$$\prod_{i=1}^{n}\prod_{j=1}^{m} log(Pr_{ij}\mathbb{1}[j = j*]) = \prod_{i=1}^{n}\prod_{j=1}^{m} log(\frac{exp((X_{ij} - X_{i1})\beta + (\alpha_j - \alpha_1))}{\sum_{1}^{m} exp((X_{ik} - X_{i1})\beta + (\alpha_k - \alpha_1))}$$

```r
#Likelihood function---------------------------------------
data = data
cl_ll = function(beta){
  x = data[, 3:12] - data[, 3]# set price 1 as reference and substract price 1 from all prices
  b = beta[1] # constant beta
  alpha = beta[2:11] # alternative-specific constant
  alpha[1] = 0  # set alpha_1 to zero
  x_beta = x * b
  alpha_choice = matrix(nrow = nrow(data), ncol = 1)
  x_beta_j = matrix(nrow = nrow(data), ncol = 1)
  alpha_t = matrix(rep(t(alpha), times = nrow(data)), ncol = ncol(t(alpha)), byrow = T)

  for (i in 1: nrow(data)){
    jstar  = data[i, "choice"]
    alpha_j = alpha[jstar]
    alpha_choice[i] = alpha_j
    x_beta_j[i] = x_beta[i, jstar]
  }
  numerator = exp(x_beta_j + alpha_choice)
  xbetak =  exp(x_beta + alpha_t)
  denominator = rowSums(xbetak)
  pr = numerator / denominator
  ll = log(pr)
  cl_ll = -sum(ll)
}

# Maxmize log-likelihood function
fit_cl = nlm( f=cl_ll, p = c(rep(0, times = 11)))
```

```
## Warning in nlm(f = cl_ll, p = c(rep(0, times = 11))): NA/Inf replaced by
## maximum positive value
```

```
par_cl = fit_cl$estimate
print(par_cl)
```

```
##  [1] -6.6565906  0.0000000 -0.9543061  1.2969786 -1.7173341 -2.9040074
##  [7] -1.5153149  0.2517637  1.4648579  2.3575174 -3.8965934
```

```
# Interpret the coefficient on price--------------------------------
# For one unit increase in price, we would expect the household's utility to decrease. Compare to the r
```

## Multinomial Logit Model

Household $i$'s probability of choosing product $j$ is modelled as:

$$Pr_{ij} = \frac{X_i\beta_j}{\sum_{l=1}^{m} exp(X_i\beta_l)},$$

where $X_i$ is a batter of household characteristics including income, family size, college, white collar, retired.

The likelihood function is

$$L = \prod_{i=1}^{n}\prod_{j=1}^{m} Pr_{ij}\nVdash[j = j*]$$

where $j*$ is the actual choice made by $i$. Log-likelihood is thus:

$$\prod_{i=1}^{n}\prod_{j=1}^{m} log(Pr_{ij}\nVdash[j = j*]) = \prod_{i=1}^{n}\prod_{l=1}^{m} log(\frac{X_i\beta_j}{\sum_{l=1}^{m} exp(X_i\beta_l)})$$

```
#Likelihood function ------------------------------

library(dummies)
indicator = dummy("choice", data = data)

x_i = as.matrix(data[, 13:19])

ll_mn = function(beta){
  beta = matrix(beta, nrow = 7, byrow = T)
  beta[, 1] = 0
  x_i_beta_j = x_i %*% beta
  ex = exp(x_i_beta_j)
  pr = t((apply(ex, 1, function(x) x / sum(x))))
  pr_choice = pr * indicator
  pr_choice = rowSums(pr_choice)
  ll_mn =  -sum(log(pr_choice))
  return(ll_mn)
}

fit_mn = nlm(f = ll_mn, p = c(rep(0, times = 70)))
```

```
## Warning in nlm(f = ll_mn, p = c(rep(0, times = 70))): NA/Inf replaced by
## maximum positive value

## Warning in nlm(f = ll_mn, p = c(rep(0, times = 70))): NA/Inf replaced by
## maximum positive value
## Warning in nlm(f = ll_mn, p = c(rep(0, times = 70))): NA/Inf replaced by
## maximum positive value
```

```
par_mn = data.frame(matrix(fit_mn$estimate, nrow = 7, byrow = T))
names(par_mn) = names(data[, 3:12])
rownames(par_mn) = names(data[, 13:19])
print(par_mn)
```

```
##           1            2            3            4           5           6
## Income    0 -0.006925862  0.009671438 -0.006315293 -0.02361269  0.02356806
## Fs3_4     0 -0.036606666 -0.280109407 -0.216251103  0.08491207 -0.81280006
## Fs5.      0 -0.043077659  0.890495371  0.214155806  0.82036114  1.64608041
## Fam_Size  0 -0.151457135 -0.944256651 -0.118343410 -0.26565162 -0.98387209
## college   0 -0.021024479  0.672493020 -0.224677815 -0.59200519  0.03671618
## whtcollar 0 -0.157741793  0.291724888 -0.180505271  0.19667654 -0.85310989
## retired   0 -0.239493061  0.405603861 -0.848327263 -1.09163071 -0.84061789
##                      7           8           9          10
## Income    -0.008476019  0.02365465  0.02362685 -0.09195729
## Fs3_4     -0.287023250  0.46739417 -0.23307792 -0.60027417
## Fs5.      -0.408256182  0.13572344  0.30596702  1.18048478
## Fam_Size  -0.282609619 -0.66714406 -0.94160813 -0.39004958
## college    0.084720499 -0.69076944 -0.47315994  0.13025132
## whtcollar -0.240665018 -0.84543251  0.31068080  0.15454283
## retired   -1.166186616 -2.04832928  0.04361183 -1.21287496
```

```
# Check with package
library(nnet)
```

```
## Warning: package 'nnet' was built under R version 3.2.3
```

```
function_mn = multinom(choice ~ Income+ Fs3_4+Fs5. + Fam_Size + college+whtcollar + retired, data = data
```

```
## # weights:  90 (72 variable)
## initial  value 10292.555366
## iter  10 value 9156.906000
## iter  20 value 8550.104966
## iter  30 value 8079.758725
## iter  40 value 7926.745864
## iter  50 value 7893.398492
## iter  60 value 7888.544409
## iter  70 value 7887.239992
## iter  80 value 7887.187733
## final  value 7887.187603
## converged
```

```
summary(function_mn)
```

```
## Call:
## multinom(formula = choice ~ Income + Fs3_4 + Fs5. + Fam_Size +
##     college + whtcollar + retired, data = data)
##
## Coefficients:
##    (Intercept)       Income        Fs3_4       Fs5.    Fam_Size
## 2    -1.023310 -0.002531375 -0.002245063 -0.2966177  0.04639359
## 3    -2.495236  0.024771587 -0.030582443  0.9127209 -0.52081828
## 4    -1.797739  0.002118081 -0.568880937 -0.7972941  0.34514970
## 5    -3.465217 -0.010052627 -0.334412840 -1.1650428  0.57992815
## 6    -2.935217  0.037420766  1.019854763  4.2798080 -0.99479452
## 7    -1.227635 -0.006019155 -0.679300195 -1.6420707  0.10246171
## 8    -2.458955  0.028944495 -0.486879122 -1.8120109  0.10598343
## 9    -1.266558  0.031253591  0.035874660  0.8870389 -0.75219226
## 10   -7.515130 -0.004249844  0.518301235  1.8200729  0.21607403
##        college   whtcollar    retired
## 2   0.04600570 -0.01955702  0.2371086
## 3   0.55488267  0.63357157  1.6984639
## 4  -0.20899804  0.04760796 -0.2187491
## 5  -0.37334352  0.72383328  0.2673713
## 6   0.06880232 -0.01798642  1.3370857
## 7   0.08641643 -0.04385822 -0.7943795
## 8  -0.44905499 -0.32061304 -1.0790889
## 9  -0.35581231  0.39716811  0.5785268
## 10 -0.08348585  2.37696749  1.0662936
##
## Std. Errors:
##    (Intercept)      Income     Fs3_4      Fs5.   Fam_Size   college
## 2    0.2099543 0.003466082 0.2014688 0.3752161 0.09414844 0.1026732
## 3    0.3383949 0.004247611 0.3231384 0.6190556 0.16118246 0.1545176
## 4    0.2219817 0.003419858 0.2117384 0.3845125 0.09670633 0.1118484
## 5    0.2970449 0.004930358 0.2785862 0.4938979 0.11892843 0.1468383
## 6    0.5622723 0.005411284 0.5399908 0.9717965 0.26155636 0.2860727
## 7    0.2818299 0.004814675 0.2713162 0.5405100 0.13198957 0.1375662
## 8    0.3547588 0.004171260 0.3271016 0.6871711 0.16127030 0.1800932
## 9    0.3274912 0.004057216 0.3145008 0.6517167 0.16048583 0.1712182
## 10   1.0680249 0.012288568 0.8520103 1.2456279 0.29289885 0.3998634
##    whtcollar   retired
## 2  0.1039874 0.1346941
## 3  0.1875123 0.2015068
## 4  0.1092279 0.1556754
## 5  0.1473156 0.2033092
## 6  0.3295017 0.3625133
## 7  0.1398639 0.1976614
## 8  0.1692255 0.2753622
## 9  0.1882916 0.2039709
## 10 0.7624902 0.5910207
##
## Residual Deviance: 15774.38
## AIC: 15918.38
```

```
# Interpret coefficien on income -------------------------------------
# Compared to reference product, higher income yields households product 2 yields households lower util
```

## Marginal Effects

```
# Conditional logit------------------

# Calculate probability for each i and j
x = data[, 3:12] - data[, 3]
b = par_cl[1]
alpha = par_cl[2:11]
```

```r
x_beta = x * b
alpha_choice = matrix(nrow = nrow(data), ncol = 1)
x_beta_j = matrix(nrow = nrow(data), ncol = 1)
alpha_t = matrix(rep(t(alpha), times = nrow(data)), ncol = ncol(t(alpha)), byrow = T)
xbetak =  exp(x_beta + alpha_t)
denominator = rowSums(xbetak)
pr_ij = as.matrix(xbetak/denominator)
pr = t(pr_ij) %*% pr_ij * (-b)
a = matrix(rep(colSums(pr_ij) * b,10), ncol=10 )
a = a * diag(10)

me_cl = data.frame((pr + a)/nrow(data))
print(me_cl)
```

```
##              X1           X2           X3          X4           X5
## 1  -1.28526906  0.295370795  0.120711900  0.29508412  0.156227495
## 2   0.29537079 -0.745429040  0.055079933  0.13345281  0.072824647
## 3   0.12071190  0.055079933 -0.337453813  0.05054413  0.030281218
## 4   0.29508412  0.133452809  0.050544129 -0.71266485  0.064015927
## 5   0.15622750  0.072824647  0.030281218  0.06401593 -0.428082220
## 6   0.03732038  0.016725820  0.007104638  0.01655091  0.008748605
## 7   0.15359586  0.069270843  0.029268647  0.06374367  0.037947887
## 8   0.09929335  0.045205906  0.019664358  0.03926138  0.025089627
## 9   0.11082171  0.050700063  0.021754537  0.04415429  0.028520040
## 10  0.01684346  0.006798224  0.003044452  0.00585762  0.004426773
##                X6           X7           X8           X9           X10
## 1   0.0373203777  0.153595860  0.099293347  0.110821713  0.0168434590
## 2   0.0167258197  0.069270843  0.045205906  0.050700063  0.0067982244
## 3   0.0071046380  0.029268647  0.019664358  0.021754537  0.0030444522
## 4   0.0165509100  0.063743674  0.039261382  0.044154286  0.0058576197
## 5   0.0087486052  0.037947887  0.025089627  0.028520040  0.0044267729
## 6  -0.1073218508  0.008537721  0.005430073  0.006113580  0.0007901258
## 7   0.0085377214 -0.420292978  0.025792624  0.027921746  0.0042139745
## 8   0.0054300734  0.025792624 -0.282460043  0.019789215  0.0029335105
## 9   0.0061135797  0.027921746  0.019789215 -0.313057324  0.0032821436
## 10  0.0007901258  0.004213974  0.002933510  0.003282144 -0.0481902824
```

```r
# Multinomial logit-----------------------------------------
x_i = as.matrix(data[, c("Income", "Fs3_4", "Fs5.",  "college", "whtcollar", "retired")])
x_i = as.matrix(cbind(x_i, rep(1, times = nrow(data))))
beta = matrix(fit_mn$estimate, nrow = 7, byrow = T)
x_i_beta_j = x_i %*% beta
ex = exp(x_i_beta_j)
pr = t((apply(ex, 1, function(x) x / sum(x))))
beta_income = matrix(beta[1, ])

beta_bar = pr %*% beta_income
beta_bar_large = matrix(rep(beta_bar, 10), ncol = 10 )
beta_j = matrix(rep(t(beta_income)), nrow(data), byrow = T, ncol = 10)
me_mn = data.frame(colSums(pr * (beta_j - beta_bar_large))/nrow(data))
names(me_mn) = "ME_Income"
row.names(me_mn) = names(data[, 3:12])
print(me_mn)
```

```
##        ME_Income
## 1  -0.0011435296
## 2  -0.0012959421
## 3   0.0005306780
## 4  -0.0005777989
## 5  -0.0007548620
## 6   0.0011693613
## 7  -0.0004656658
## 8   0.0004134450
## 9   0.0027828282
## 10 -0.0006585140
```

```
# Interpret marginal effects
# For conditional logit, if the price of product 1 increases, we would expect households to be less lik
# For multinomial logit, if the income of household increases, we would expect households to be less li
```

## Mixed Logit and IIA

```
# Mixed logit model-------------------------------------------------------
x_ij = data[, 3:12]
w_i = as.matrix(data[, c("Income", "Fs3_4", "Fs5.",  "college", "whtcollar", "retired")])
w_i = as.matrix(cbind(rep(1, times = nrow(data)), w_i))
indicator = indicator


# Likelihood function
ml_ll = function(b){
  beta = b[1]
  gamma = matrix(b[2:71], nrow = 7, ncol =10,  byrow = T)
  gamma[, 1] = 0
  x_i_j_beta = x_ij * beta
  w_i_r_j = w_i %*% gamma
  num = rowSums(exp((x_i_j_beta + w_i_r_j) * indicator))
  denom = rowSums(exp((x_i_j_beta + w_i_r_j)))
  pr = num / denom
  ml_ll = -sum(log(pr))
  return(ml_ll)
}

# Maximize likelihood
fit_ml = nlm(f = ml_ll, p = rep(0, times = 71))
```

```
## Warning in nlm(f = ml_ll, p = rep(0, times = 71)): NA/Inf replaced by
## maximum positive value

## Warning in nlm(f = ml_ll, p = rep(0, times = 71)): NA/Inf replaced by
## maximum positive value

## Warning in nlm(f = ml_ll, p = rep(0, times = 71)): NA/Inf replaced by
## maximum positive value

## Warning in nlm(f = ml_ll, p = rep(0, times = 71)): NA/Inf replaced by
## maximum positive value
## Warning in nlm(f = ml_ll, p = rep(0, times = 71)): NA/Inf replaced by
## maximum positive value

## Warning in nlm(f = ml_ll, p = rep(0, times = 71)): NA/Inf replaced by
## maximum positive value

## Warning in nlm(f = ml_ll, p = rep(0, times = 71)): NA/Inf replaced by
## maximum positive value

## Warning in nlm(f = ml_ll, p = rep(0, times = 71)): NA/Inf replaced by
## maximum positive value

## Warning in nlm(f = ml_ll, p = rep(0, times = 71)): NA/Inf replaced by
## maximum positive value

## Warning in nlm(f = ml_ll, p = rep(0, times = 71)): NA/Inf replaced by
## maximum positive value

## Warning in nlm(f = ml_ll, p = rep(0, times = 71)): NA/Inf replaced by
## maximum positive value

## Warning in nlm(f = ml_ll, p = rep(0, times = 71)): NA/Inf replaced by
## maximum positive value

## Warning in nlm(f = ml_ll, p = rep(0, times = 71)): NA/Inf replaced by
## maximum positive value

## Warning in nlm(f = ml_ll, p = rep(0, times = 71)): NA/Inf replaced by
## maximum positive value

## Warning in nlm(f = ml_ll, p = rep(0, times = 71)): NA/Inf replaced by
## maximum positive value

## Warning in nlm(f = ml_ll, p = rep(0, times = 71)): NA/Inf replaced by
## maximum positive value

## Warning in nlm(f = ml_ll, p = rep(0, times = 71)): NA/Inf replaced by
## maximum positive value

## Warning in nlm(f = ml_ll, p = rep(0, times = 71)): NA/Inf replaced by
## maximum positive value

## Warning in nlm(f = ml_ll, p = rep(0, times = 71)): NA/Inf replaced by
## maximum positive value

## Warning in nlm(f = ml_ll, p = rep(0, times = 71)): NA/Inf replaced by
## maximum positive value

## Warning in nlm(f = ml_ll, p = rep(0, times = 71)): NA/Inf replaced by
## maximum positive value

## Warning in nlm(f = ml_ll, p = rep(0, times = 71)): NA/Inf replaced by
## maximum positive value
## Warning in nlm(f = ml_ll, p = rep(0, times = 71)): NA/Inf replaced by
## maximum positive value

## Warning in nlm(f = ml_ll, p = rep(0, times = 71)): NA/Inf replaced by
## maximum positive value

## Warning in nlm(f = ml_ll, p = rep(0, times = 71)): NA/Inf replaced by
## maximum positive value

## Warning in nlm(f = ml_ll, p = rep(0, times = 71)): NA/Inf replaced by
## maximum positive value

## Warning in nlm(f = ml_ll, p = rep(0, times = 71)): NA/Inf replaced by
## maximum positive value

## Warning in nlm(f = ml_ll, p = rep(0, times = 71)): NA/Inf replaced by
## maximum positive value

## Warning in nlm(f = ml_ll, p = rep(0, times = 71)): NA/Inf replaced by
## maximum positive value

## Warning in nlm(f = ml_ll, p = rep(0, times = 71)): NA/Inf replaced by
## maximum positive value

## Warning in nlm(f = ml_ll, p = rep(0, times = 71)): NA/Inf replaced by
## maximum positive value

## Warning in nlm(f = ml_ll, p = rep(0, times = 71)): NA/Inf replaced by
## maximum positive value

## Warning in nlm(f = ml_ll, p = rep(0, times = 71)): NA/Inf replaced by
## maximum positive value

## Warning in nlm(f = ml_ll, p = rep(0, times = 71)): NA/Inf replaced by
## maximum positive value

## Warning in nlm(f = ml_ll, p = rep(0, times = 71)): NA/Inf replaced by
## maximum positive value

## Warning in nlm(f = ml_ll, p = rep(0, times = 71)): NA/Inf replaced by
## maximum positive value

## Warning in nlm(f = ml_ll, p = rep(0, times = 71)): NA/Inf replaced by
## maximum positive value

## Warning in nlm(f = ml_ll, p = rep(0, times = 71)): NA/Inf replaced by
## maximum positive value

## Warning in nlm(f = ml_ll, p = rep(0, times = 71)): NA/Inf replaced by
## maximum positive value

## Warning in nlm(f = ml_ll, p = rep(0, times = 71)): NA/Inf replaced by
## maximum positive value
## Warning in nlm(f = ml_ll, p = rep(0, times = 71)): NA/Inf replaced by
## maximum positive value

## Warning in nlm(f = ml_ll, p = rep(0, times = 71)): NA/Inf replaced by
## maximum positive value

## Warning in nlm(f = ml_ll, p = rep(0, times = 71)): NA/Inf replaced by
## maximum positive value

## Warning in nlm(f = ml_ll, p = rep(0, times = 71)): NA/Inf replaced by
## maximum positive value

## Warning in nlm(f = ml_ll, p = rep(0, times = 71)): NA/Inf replaced by
## maximum positive value

## Warning in nlm(f = ml_ll, p = rep(0, times = 71)): NA/Inf replaced by
## maximum positive value

## Warning in nlm(f = ml_ll, p = rep(0, times = 71)): NA/Inf replaced by
## maximum positive value

## Warning in nlm(f = ml_ll, p = rep(0, times = 71)): NA/Inf replaced by
## maximum positive value

## Warning in nlm(f = ml_ll, p = rep(0, times = 71)): NA/Inf replaced by
## maximum positive value

## Warning in nlm(f = ml_ll, p = rep(0, times = 71)): NA/Inf replaced by
## maximum positive value

## Warning in nlm(f = ml_ll, p = rep(0, times = 71)): NA/Inf replaced by
## maximum positive value

## Warning in nlm(f = ml_ll, p = rep(0, times = 71)): NA/Inf replaced by
## maximum positive value

## Warning in nlm(f = ml_ll, p = rep(0, times = 71)): NA/Inf replaced by
## maximum positive value

## Warning in nlm(f = ml_ll, p = rep(0, times = 71)): NA/Inf replaced by
## maximum positive value

## Warning in nlm(f = ml_ll, p = rep(0, times = 71)): NA/Inf replaced by
## maximum positive value

## Warning in nlm(f = ml_ll, p = rep(0, times = 71)): NA/Inf replaced by
## maximum positive value

## Warning in nlm(f = ml_ll, p = rep(0, times = 71)): NA/Inf replaced by
## maximum positive value

## Warning in nlm(f = ml_ll, p = rep(0, times = 71)): NA/Inf replaced by
## maximum positive value
## Warning in nlm(f = ml_ll, p = rep(0, times = 71)): NA/Inf replaced by
## maximum positive value

## Warning in nlm(f = ml_ll, p = rep(0, times = 71)): NA/Inf replaced by
## maximum positive value

## Warning in nlm(f = ml_ll, p = rep(0, times = 71)): NA/Inf replaced by
## maximum positive value

## Warning in nlm(f = ml_ll, p = rep(0, times = 71)): NA/Inf replaced by
## maximum positive value

## Warning in nlm(f = ml_ll, p = rep(0, times = 71)): NA/Inf replaced by
## maximum positive value

## Warning in nlm(f = ml_ll, p = rep(0, times = 71)): NA/Inf replaced by
## maximum positive value

## Warning in nlm(f = ml_ll, p = rep(0, times = 71)): NA/Inf replaced by
## maximum positive value

## Warning in nlm(f = ml_ll, p = rep(0, times = 71)): NA/Inf replaced by
## maximum positive value

## Warning in nlm(f = ml_ll, p = rep(0, times = 71)): NA/Inf replaced by
## maximum positive value

## Warning in nlm(f = ml_ll, p = rep(0, times = 71)): NA/Inf replaced by
## maximum positive value

## Warning in nlm(f = ml_ll, p = rep(0, times = 71)): NA/Inf replaced by
## maximum positive value

## Warning in nlm(f = ml_ll, p = rep(0, times = 71)): NA/Inf replaced by
## maximum positive value

## Warning in nlm(f = ml_ll, p = rep(0, times = 71)): NA/Inf replaced by
## maximum positive value
```

```r
par_ml = fit_ml$estimate
par_ml = data.frame(matrix(fit_ml$estimate[2:71], nrow = 7, byrow = T))
par_ml = rbind(par_ml, rep(fit_ml$estimate[1], times = 10, byrow = T))
names(par_ml) = names(data[, 3:12])
rownames(par_ml) = c(names(data[, 13:19]), "price")


#Test IIA------------------------------------------------

# Restrict data to choices not equal to 1
d_res = data %>%
        filter(choice!=1)
x_ij_res = d_res[, 4:12]
w_i_res = as.matrix(d_res[, c("Income", "Fs3_4", "Fs5.",  "college", "whtcollar", "retired")])
w_i_res = as.matrix(cbind(w_i_res, rep(1, times = nrow(d_res))))
indicator = dummy("choice", data = d_res)

# LikelihooD function for the restricted model
ml_ll_res = function(b){
  beta = b[1]
  gamma = matrix(b[2:64], nrow = 7, ncol = 9,  byrow = T)
  gamma[, 1] = 0
  x_i_j_beta = x_ij_res * beta
  w_i_r_j = w_i_res %*% gamma
  num = rowSums(exp(x_i_j_beta + w_i_r_j) * indicator)
  denom = rowSums(exp((x_i_j_beta + w_i_r_j)))
  pr = num / denom
  ml_ll_res = -sum(log(pr))
  return(ml_ll_res)
}


fit_ml_res = nlm(f = ml_ll_res, p = rep(0, times = 64))
```

```
## Warning in nlm(f = ml_ll_res, p = rep(0, times = 64)): NA/Inf replaced by
## maximum positive value

## Warning in nlm(f = ml_ll_res, p = rep(0, times = 64)): NA/Inf replaced by
## maximum positive value

## Warning in nlm(f = ml_ll_res, p = rep(0, times = 64)): NA/Inf replaced by
## maximum positive value
```

```
par_ml_res = fit_ml_res$estimate
par_ml_res = data.frame(matrix(fit_ml_res$estimate[2:71], nrow = 7, byrow = T))
par_ml_res = rbind(par_ml_res, rep(fit_ml_res$estimate[1], times = 10, byrow = T))
names(par_ml_res) = names(data[, 4:12])
rownames(par_ml_res) = c(names(d_res[, 13:19]), "price")

# Compute test statistics
beta_f = fit_ml$estimate
beta_r = fit_ml_res$estimate
L_f = ml_ll(beta_f)
L_r = ml_ll_res(beta_r)
MTT = -2 * (L_f - L_r)
library(chi)
pchi(MTT, 57, lower.tail = F)
```

```
## [1] 0
```

```
# Therefore, IIA property is violated.
```