Web Appendix
Multinomial Logit Models in Marketing – From Fundamentals to State-of-the-Art

```
# ==========================================================================================
# Accompanying R code to:
# Elshiewy, Guhl, and Boztug (2017).
# Multinomial Logit Models in Marketing – From Fundamentals to State-of-the-Art.
# Published in: Marketing ZFP - Journal of Research and Management, Volume 39, Issue 3.


# Load packages ============================================================================
# The following packages are necessary to run the script. In case these packages are not
# already installed, please run install.packages("package-name") first for each package.
library(mlogit)
library(data.table)
library(ggplot2)
library(psych)
library(mvtnorm)
library(bayesm)
library(gmnl)


# Functions ================================================================================
# The following functions are necessary for some of the computational steps after the
# estimation (e.g., computing elasticities).

# This function computes (heterogeneous) probabilities and market shares for a
# (heterogeneous) MNL model given draws from the estimated population distribution.
mnlProb <- function(x, betai) {
  u <- x %*% t(betai)
  num <- exp(u)
  tprobs <- t(num) / colSums(num)
  probs <- t(tprobs)
  shares <- rowMeans(probs)
  res <- list(shares = shares,
              probs = probs)
  return(res)
}


# This function computes the matrix of first derivatives for a (heterogeneous) MNL model
# given (heterogeneous) preferences and corresponding probabilities.
mnlDeriv <- function(betaxi, probi) {
  nJ <- dim(probi)[1]
  nD <- dim(probi)[2]
  betaxim <- matrix(betaxi, nJ, nD, byrow = TRUE)
  x <- rep(1, nJ)
  eta <-  -(1 / x) %*% t(rep(1, nJ)) * tcrossprod(betaxim * probi, probi) / nD
  diag(eta) <- (1 / x) * rowSums(betaxim * probi * (1 - probi)) / nD
  return(eta)
}


# This function computes an elasticity matrix. "coef" specifies the position of the
# variable of interest in x and beta. (Default is coef = 4, here: price)
mnlElast <- function(x, betai, coef = 4) {
  nJ <- nrow(x)
  probs <- mnlProb(x, betai)
  shares <- probs$shares
  shares <- matrix(shares, nJ, nJ, byrow = TRUE)
  eta <- x[, coef] / shares * mnlDeriv(betai[, coef], probs$probs)
  return(eta)
}
```

```r
# This is a convenience function for computing elasticities for several choice sets
# "coef" specifies the position of the variable of interest in data and betai.
# (Default is coef = 4, here: price)
computeElast <- function(data, betai, coef = 4) {
  E_chid <- sapply(data, FUN = mnlElast,
                   betai = betai, coef = coef,
                   simplify = "array")

  E <- apply(E_chid, c(1, 2), mean)
  E[c(1, 2, 4, 3), c(1, 2, 4, 3)] # reorder matrix (delete this line for own analysis!)
}

# This functions reshapes and summarises output of bayesm
reshapeBayesmDraws <- function(out, post_index = NULL, coef_names = NULL) {

  C <- dim(out$nmix$probdraw)[2] #  number of components
  I <- dim(out$betadraw)[1] # number of individuals
  K <- dim(out$betadraw)[2] # number of parameters
  R <- dim(out$betadraw)[3] # number of draws

  if (is.null(post_index)) post_index <- 1:R
  if (is.null(coef_names)) coef_names <- paste0("par_", 1:K)

  # pop draws
  sdMat <- bMat <- array(NA, dim = c(R, K))
  corArray <- sigmaArray <- array(NA, dim = c(K, K, R))
  muArray <- array(NA, dim = c(R, K, C))
  sigArray <- array(NA, dim = c(K, K, R, C))
  for (r in 1:R) {
    moms <- momMix(out$nmix$probdraw[r,, drop = FALSE], out$nmix$compdraw[r])
    sigmaArray[,, r] <- moms$sigma
    corArray[,, r] <- moms$corr
    sdMat[r,] <- moms$sd
    bMat[r,] <- moms$mu
    for (c in 1:C) {
      rTemp <- out$nmix$compdraw[[r]][[c]]$rooti
      sigArray[,, r, c] <- crossprod(chol(chol2inv(chol(tcrossprod(rTemp)))))
      muArray[r,, c] <- out$nmix$compdraw[[r]][[c]]$mu
    }
  }
  colnames(sigArray) <- rownames(sigArray) <- coef_names
  colnames(sigmaArray) <- rownames(sigmaArray) <- coef_names
  colnames(corArray) <- rownames(corArray) <- coef_names
  colnames(sdMat) <- colnames(bMat) <- colnames(muArray) <- coef_names

  # ind draws
  betaMat <- out$betadraw
  colnames(betaMat) <- coef_names

  return(list(b = bMat[post_index,],
              sd = sdMat[post_index,],
              Sigma = sigmaArray[,, post_index],
              Cor = corArray[,, post_index],
              beta = betaMat[,, post_index],
              loglike = out$loglike[post_index],
              muArray = muArray[post_index,,],
              sigArray = sigArray[,, post_index,],
              probdraw = out$nmix$probdraw[post_index,]))
}

trim <- function(x, prob = 0.01) {
  subset(x, x > quantile(x, probs = prob) & x < quantile(x, probs = 1 - prob))
}

# Load data ====================================================================
data(Cracker) # Cracker data of Jain et al. (1994); included in the mlogit package

cracker <- as.data.table(Cracker)
cracker[, cs := 1:.N, by = id] # add choice situation for each consumer
```

```r
# Descriptives stats =====================================================================

# number of consumers
# (we exclude the first choice for each consumer later to initialize the 'lastchoice' variable)
cracker[cs != 1, uniqueN(id)]

# number of choices for each consumer
n_choice_id <- cracker[cs != 1, .N, by = id]
describe(n_choice_id[, .(N)], fast = TRUE)
# number of choices
(n <- n_choice_id[, sum(N)])

# choices
cracker[cs != 1, table(choice)]

# choice shares
cracker[cs != 1, round(table(choice) / .N, 3)]

# prices (in $-Cent / oz => divided by 100 => $ / oz, see Jain et al. 1994)
describe(cracker[cs != 1, 10:13] / 100, fast = TRUE)

# display (fraction of purchase occasions on display promotion for each brand)
describe(cracker[cs != 1, 2:5], fast = TRUE)

# feature (fraction of purchase occasions on which each brand was featured)
describe(cracker[cs != 1, 6:9], fast = TRUE)


# Data prep ==============================================================================

# build last choice variable and delete first choice (=> initialisation)
last_choice <- cracker[, .(id, cs = cs + 1L, lastchoice = choice)]
cracker <- merge(cracker, last_choice, by = c("id", "cs"))

# reshape data to long format
cracker_long <- mlogit.data(cracker, id = "id", choice = "choice", varying = c(3:14),
                            shape = "wide", sep = ".")

# transform lastchoice variable into dummy
cracker_long$lastchoice <- as.integer(cracker_long$last == cracker_long$alt)

# divide price by 100 => $ / oz
cracker_long$price <- cracker_long$price / 100

# make sure "private" is reference alternative in analyses
attr(cracker_long, "index")[["alt"]] <- relevel(attr(cracker_long, "index")[["alt"]],
                                                "private")

# prep/reshape data for Bayesian estimation
M <- model.matrix(mFormula(choice ~ price + feat + disp + lastchoice),
                  data = cracker_long)
coef_names <- colnames(M)
cracker_hb <- NULL
for (i in unique(cracker_long$id)) {
  cracker_hb[[i]] <- list(y = match(subset(cracker_long, id == i & (choice))[,"alt"],
                                    cracker_long[1:4, "alt"]),
                          X = M[which(cracker_long$id == i),])
}

# prep/reshape data for elasticity computation
cracker_elast <- NULL
for (i in unique(cracker_long$chid)) {
  M_i <- M[cracker_long$chid == i,]
  rownames(M_i) <- c("keebler", "nabisco", "private", "sunshine")
  cracker_elast[[i]] <- M_i
}
```

```
# DCM estimation =========================================================================
# MNL ------------------------------------------------------------------------------------

# benchmark model: brand intercepts only
mnl0 <- gmnl(choice ~ 1, data = cracker_long)
(ll_0 <- logLik(mnl0))

# MNL model (mnl0 + price, feature, display and lastchoice as covariates)
mnl <- gmnl(choice ~ price + feat + disp + lastchoice, data = cracker_long)

# estimates
round(summary(mnl)$CoefTable, 3)

# LL
logLik(mnl)

# McFadden R^2
round(c(1 - logLik(mnl) / ll_0), 3)

# Price elasticity (average over all observations)
round(computeElast(cracker_elast, t(coef(mnl))), 3)


# Nested MNL ----------------------------------------------------------------------------
nl <- mlogit(choice ~ price + feat + disp + lastchoice, data = cracker_long,
             nests = list(national_brands = c("keebler", "nabisco", "sunshine"),
                          store_brand = c("private")), un.nest.el = TRUE)

# estimates
round(summary(nl)$CoefTable, 3)

# LL
logLik(nl)

# McFadden R^2
round(c(1 - logLik(nl) / ll_0), 3)


# LC MNL --------------------------------------------------------------------------------
# We use different starting values here because LC MNL models can have several local optima.
# For each number of segments (2-6), we estimate the model using ten starting values
# and kept the run with the highest LL-value.
# The minimum BIC value was used to determine the optimal number of latent classes (here 5).
lc5 <- gmnl(choice ~ price + feat + disp + lastchoice | 1 | 0 | 0 | 1,
            start = c(-3.3, -2, -3, 1.3, 0.1, 0.5, 1, 1, 4, 0.3, -2, 1, 0.2,
                      1, -0.5, 2, -0.3, -11, -0.4, 0.5, 0.4, 1.5, 2, 1, -5, 1,
                      0.5, 0.7, 7, 6.5, 5.5, -8, 0, 0, 1, 1, 0.4, 0.4, -0.3),
            data = cracker_long, print.level = 1, model = "lc", Q = 5, panel = TRUE)

# estimates
round(summary(lc5)$CoefTable, 3)

# LL
logLik(lc5)

# McFadden R^2
round(c(1 - logLik(lc5) / ll_0), 3)

# latent class sizes (prior probs)
coef_class <- c(0, coef(lc5)[36:39])
names(coef_class) <- as.character(1:5)
prior_probs <- exp(coef_class) / sum(exp(coef_class))
round(prior_probs, 3)

# crisp segmentation based on max posterior probs
post_probs <- lc5$Qir
colnames(post_probs) <- as.character(1:5)
```

```r
# 3 consumers as example
round(post_probs[c(7, 24, 33), ], 3)

# #consumer assigned to latent classes
class_assignment <- apply(post_probs, 1, which.max)
table(class_assignment)

# average posterior membership probability of assigned consumers
round(tapply(post_probs[cbind(1:136, class_assignment)], class_assignment, mean), 3)

# individual coefficients
lc5_beta_i <- data.table(effect.gmnl(lc5)$mean)
ggplot(suppressWarnings(melt(lc5_beta_i)), aes(x = value)) +
  geom_histogram(bins = 10) + theme_minimal() + ggtitle("LC (5)") +
  facet_wrap(~variable, ncol = 4, scales = "free")

# Price elasticities
round(computeElast(cracker_elast, t(lc5$bi[1,])), 3) # latent class 1
round(computeElast(cracker_elast, t(lc5$bi[2,])), 3) # latent class 2
round(computeElast(cracker_elast, t(lc5$bi[3,])), 3) # latent class 3
round(computeElast(cracker_elast, t(lc5$bi[4,])), 3) # latent class 4
round(computeElast(cracker_elast, t(lc5$bi[5,])), 3) # latent class 5

# draws from the (point estimate of the) discrete population distribution
set.seed(1234)
lc5_beta_r <- lc5$bi[sample(1:5, 10000, prob = prior_probs, replace = TRUE),]
round(computeElast(cracker_elast, lc5_beta_r), 3) # aggregated (over all classes)


# M-MNL ---------------------------------------------------------------------------------
mmnl <- gmnl(choice ~ price + feat + disp + lastchoice | 1, data = cracker_long,
             model = "mixl", correlation = TRUE, halton = NA, R = 1000,
             panel = TRUE, tol = 1e-12, print.level = 1,
             ranp = c(`keebler:(intercept)` = "n",
                      `nabisco:(intercept)` = "n",
                      `sunshine:(intercept)` = "n",
                      price = "n", feat = "n", disp = "n", lastchoice = "n"))

# means of random parameters
round(summary(mmnl)$CoefTable[1:7,], 3)

# standard deviations of the random parameters
vcov(mmnl, what = "ranp", type = "sd", se = TRUE, digits = 3)

# correlations of random parameters
round(cor.gmnl(mmnl), 3)

# LL
logLik(mmnl)

# McFadden R^2
round(c(1 - logLik(mmnl) / ll_0), 3)

# individual coefficients
mmnl_beta_i <- data.table(effect.gmnl(mmnl)$mean)
ggplot(suppressWarnings(melt(mmnl_beta_i)), aes(x = value)) +
  geom_histogram(bins = 10) + theme_minimal() + ggtitle("MMNL") +
  facet_wrap(~variable, ncol = 4, scales = "free")

# Price elasticities
# draws from the (point estimate of the) population distribution
set.seed(1234)
mmnl_beta_r <- rmvnorm(10000, coef(mmnl)[1:7], cov.gmnl(mmnl))
round(computeElast(cracker_elast, mmnl_beta_r), 3)
```

```r
# HB MNL (normal prior) -----------------------------------------------------------------
# setup
R <- 200000 # draws
keep <- 100 # keep every 100th draw
post_index <- seq(1 + 0.5 * R / keep, R / keep) # index for posterior draws (50% warmup)

# HB estimation
set.seed(1234)
hbmnl <- rhierMnlRwMixture(Data = list(p = 4, lgtdata = cracker_hb),
                           Prior = list(ncomp = 1), # default priors, 1 component
                           Mcmc = list(R = R, keep = keep, nprint = R / 20))

# reshape draws for easier handling
hbmnl_draws <- reshapeBayesmDraws(hbmnl, post_index, coef_names)

# LMD
logMargDenNR(trim(hbmnl_draws$loglike))

# means of random parameters
round(cbind(mean = apply(hbmnl_draws$b, 2, mean),
            t(apply(hbmnl_draws$b, 2, quantile, prob = c(0.025, 0.975)))), 3)

# standard deviations of the random parameters
round(cbind(mean = apply(hbmnl_draws$sd, 2, mean),
            t(apply(hbmnl_draws$sd, 2, quantile, prob = c(0.025, 0.975)))), 3)

# correlations of random parameters
round(apply(hbmnl_draws$Cor, c(1, 2), mean), 3)

# individual coefficients
hbmnl_beta_i <- data.table(apply(hbmnl_draws$beta, c(1, 2), mean))
ggplot(suppressWarnings(melt(hbmnl_beta_i)), aes(x = value)) +
  geom_histogram(bins = 10) + theme_minimal() + ggtitle("HBMNL") +
  facet_wrap(~variable, ncol = 4, scales = "free")

# Price elasticities
# draws from the (posterior mean of the) population distribution
set.seed(1234)
hbmnl_beta_r <- rmvnorm(10000, apply(hbmnl_draws$b, 2, mean),
                        apply(hbmnl_draws$Sigma, c(1, 2), mean))
round(computeElast(cracker_elast, hbmnl_beta_r), 3)


# HB MNL (mixture of normals prior) -----------------------------------------------------
# HB estimation
set.seed(1234)
hbmnl3 <- rhierMnlRwMixture(Data = list(p = 4, lgtdata = cracker_hb),
                            Prior = list(Amu = 1/10, ncomp = 3, a = rep(5/3, 3)),
                            Mcmc = list(R = R, keep = keep, nprint = R / 20))
# we use a slightly tighter prior for 'Amu' than the default setting,
# because all variables are reasonable scaled (see Rossi et al. 2005, p. 150)

# reshape draws for easier handling
hbmnl3_draws <- reshapeBayesmDraws(hbmnl3, post_index, coef_names)

# LMD
logMargDenNR(trim(hbmnl3_draws$loglike))

# means of random parameters
round(cbind(mean = apply(hbmnl3_draws$b, 2, mean),
            t(apply(hbmnl3_draws$b, 2, quantile, prob = c(0.025, 0.975)))), 3)

# standard deviations of the random parameters
round(cbind(mean = apply(hbmnl3_draws$sd, 2, mean),
            t(apply(hbmnl3_draws$sd, 2, quantile, prob = c(0.025, 0.975)))), 3)

# correlations of random parameters
round(apply(hbmnl3_draws$Cor, c(1, 2), mean), 3)
```

```
# individual coefficients
hbmnl3_beta_i <- data.table(apply(hbmnl3_draws$beta, c(1, 2), mean))
ggplot(suppressWarnings(melt(hbmnl3_beta_i)), aes(x = value)) +
  geom_histogram(bins = 10) + theme_minimal() + ggtitle("HBMNL(3)") +
  facet_wrap(~variable, ncol = 4, scales = "free")
# scatterplot: HB MBL (3) vs HB MNL
hbmnl_1_and_3_beta_i <- rbind(hbmnl_beta_i, hbmnl3_beta_i)
hbmnl_1_and_3_beta_i[, model := rep(c("HB MNL", "HB MNL (3)"), each = 136)]
hbmnl_1_and_3_beta_i[, id := rep(1:136, 2)]


hbmnl_1_and_3_beta_i <- melt(hbmnl_1_and_3_beta_i, id.vars = c("model", "id"))
hbmnl_1_and_3_beta_i <- dcast.data.table(hbmnl_1_and_3_beta_i,
                                         variable + id ~ model, fill = FALSE)
hbmnl_1_and_3_beta_i[variable == "keebler:(intercept)", variable := "Keebler"]
hbmnl_1_and_3_beta_i[variable == "nabisco:(intercept)", variable := "Nabisco"]
hbmnl_1_and_3_beta_i[variable == "sunshine:(intercept)", variable := "Sunshine"]
hbmnl_1_and_3_beta_i[variable == "price", variable := "Price"]
hbmnl_1_and_3_beta_i[variable == "feat", variable := "Feature"]
hbmnl_1_and_3_beta_i[variable == "disp", variable := "Display"]
hbmnl_1_and_3_beta_i[variable == "lastchoice", variable := "Lastchoice"]


ggplot(hbmnl_1_and_3_beta_i,
       aes(y = `HB MNL (3)`, x = `HB MNL`)) +
  geom_point(size = 1,pch = 1) + theme_minimal() +
  geom_abline(slope = 1, intercept = 0, col = "grey20") +
  facet_wrap(~variable, ncol = 4, scales = "free") +
  theme(text = element_text(family = "serif")) +
  ylab(expression(paste(paste(beta)["i"], " HB MNL (3)"))) +
  xlab(expression(paste(paste(beta)["i"], " HB MNL")))

# Price elasticities
# draws from the (posterior mean of the) population distribution
muc <- apply(hbmnl3_draws$muArray, c(2, 3), mean)
sigc <- apply(hbmnl3_draws$sigArray, c(1, 2, 4), mean)
pvec <- apply(hbmnl3_draws$probdraw, 2, mean)
comps <- list()
for (c in 1:length(pvec)) {
  comps[[c]] <- list(mu = muc[, c], rooti = solve(chol(sigc[,,c])))
}

set.seed(1234)
hbmnl3_beta_r <- rmixture(10000, pvec, comps)$x
round(computeElast(cracker_elast, hbmnl3_beta_r), 3)


# comparison of marginal distribution of the price coefficients
price_draws <- melt(data.table(`HB MNL` = hbmnl_beta_r[, 4],
                               `HB MNL (3)` = hbmnl3_beta_r[, 4]),
                    measure.vars = c("HB MNL", "HB MNL (3)"),
                    variable.name = "model", value.name = "beta_price")

ggplot(price_draws, aes(x = beta_price, fill = model)) +
  geom_density(adjust = 2, lwd = 1, alpha = .5) + theme_minimal() +
  ylim(c(0, 0.08)) + xlim(c(-30, 20)) +
  scale_fill_grey(start = 0.2, end = 0.8) +
  theme(text = element_text(family = "serif")) +
  xlab(expression(paste(beta)["price"])) +
  theme(legend.title = element_blank())
```