Own- and Cross-Price Demand Curve Analyses for Food and Social Reinforcer

1 Preliminaries

In this section, the RStudio workspace and console panes are cleared of old output, variables, and other miscellaneous debris. Be sure that nothing to be removed is needed. Once the decks are cleared, get required packages and data files.

1.1 Options

```
# Set some global options
options(replace.assign = TRUE, width = 65, digits = 4, scipen = 4, fig.width = 4,
    fig.height = 4)
# Clear the workspace and console
rm(list = ls(all.names = TRUE))
cat("\f")
```

```
# Start timing
how_long <- Sys.time()
```

1.2 Packages

```
library(here)
library(readr)
library(dplyr)
library(ggplot2)
library(tidyverse)
library(ggpubr)
library(minpack.lm)
```

1.3 Get the Data Functions

```r
# Function
lhs <- function(x) {
    # log-like scale
    log10(0.5 * x + sqrt(0.25 * (x^2) + 1))
}
antilog <- function(y) {
    # Antilog
    (10^(2 * y) - 1)/(10^y)
}
ev <- function(y) {
    # Expected Value
    1/(100 * y)
}
# Data
Behav <- read.csv("data/data with bl mean.csv") |>
    mutate(foodr_lhs = lhs(foodr), socr_lhs = lhs(socr))
```

2 Own-price Demand Curve Fitting

The following analyses were conducted to fit the demand data with the Zero-Bounded Exponential model (Gilroy et al., 2021). It should be noted that the values of the parameters may not be exactly the same as the ones reported on the published article because of the different software algorithms were used.

2.1 Condition 1: Food own price demand

```
group_by(subset(Behav, cond == 1), subj) |>
    mutate(alpha = coef(nlsLM(foodr_lhs ~ lhs(Q_0) * (exp((-(a)/lhs(Q_0)) *
        (Q_0) * foodfr)), start = list(a = 0.0001, Q_0 = 50), control = nls.lm.control(m
        alpha_se = summary(nlsLM(foodr_lhs ~ lhs(Q_0) * (exp((-(a)/lhs(Q_0)) *
            (Q_0) * foodfr)), start = list(a = 0.0001, Q_0 = 50),
            control = nls.lm.control(maxiter = 1024)))$coef["a", "Std. Error"],
        Q0 = coef(nlsLM(foodr_lhs ~ lhs(Q_0) * (exp((-(a)/lhs(Q_0)) *
            (Q_0) * foodfr)), start = list(a = 0.0001, Q_0 = 50),
            control = nls.lm.control(maxiter = 1024)))[2], Q0_se = summary(nlsLM(foodr_l
            lhs(Q_0) * (exp((-(a)/lhs(Q_0)) * (Q_0) * foodfr)), start = list(a = 0.0001,
            Q_0 = 50), control = nls.lm.control(maxiter = 1024)))$coef["Q_0",
            "Std. Error"]) |>
    slice(n()) |>
    select(subj, cond, alpha, alpha_se, Q0, Q0_se)

## # A tibble: 4 x 6
## # Groups:   subj [4]
##    subj  cond    alpha  alpha_se    Q0 Q0_se
##   <int> <int>    <dbl>     <dbl> <dbl> <dbl>
## 1     1     1 0.000455 0.0000840  178.  50.4
## 2     2     1 0.000117 0.0000165  260.  50.0
## 3     3     1 0.000287 0.0000730  393. 154.
## 4     4     1 0.000192 0.0000356  218.  57.0
```

2.2 Condition 2: Social own price demand

```
group_by(subset(Behav, cond == 2), subj) |>
    mutate(alpha = coef(nlsLM(socr_lhs ~ lhs(Q_0) * (exp((-(a)/lhs(Q_0)) *
        (Q_0) * socfr)), start = list(a = 0.0001, Q_0 = 50), control = nls.lm.control(ma
        alpha_se = summary(nlsLM(socr_lhs ~ lhs(Q_0) * (exp((-(a)/lhs(Q_0)) *
            (Q_0) * socfr)), start = list(a = 0.0001, Q_0 = 50), control = nls.lm.contro
            "Std. Error"], Q0 = coef(nlsLM(socr_lhs ~ lhs(Q_0) * (exp((-(a)/lhs(Q_0)) *
            (Q_0) * socfr)), start = list(a = 0.0001, Q_0 = 50), control = nls.lm.contro
        Q0_se = summary(nlsLM(socr_lhs ~ lhs(Q_0) * (exp((-(a)/lhs(Q_0)) *
            (Q_0) * socfr)), start = list(a = 0.0001, Q_0 = 50), control = nls.lm.contro
            "Std. Error"]) |>
    slice(n()) |>
    select(subj, cond, alpha, alpha_se, Q0, Q0_se)

## # A tibble: 4 x 6
## # Groups:   subj [4]
##    subj  cond   alpha alpha_se    Q0 Q0_se
##   <int> <int>   <dbl>    <dbl> <dbl> <dbl>
## 1     1     2 0.00407  0.00134  36.6 22.3
## 2     2     2 0.00282  0.00105  67.9 47.1
## 3     3     2 0.0108   0.00215  15.3  5.63
## 4     4     2 0.00424  0.00130  48.4 29.4
```

2.3 Condition 3

```
# Food own price demand
group_by(subset(Behav, cond == 3), subj) |>
    mutate(alpha = coef(nlsLM(foodr_lhs ~ lhs(Q_0) * (exp((-(a)/lhs(Q_0)) *
        (Q_0) * foodfr)), start = list(a = 0.0001, Q_0 = 50), control = nls.lm.control(m
        alpha_se = summary(nlsLM(foodr_lhs ~ lhs(Q_0) * (exp((-(a)/lhs(Q_0)) *
            (Q_0) * foodfr)), start = list(a = 0.0001, Q_0 = 50),
            control = nls.lm.control(maxiter = 1024)))$coef["a", "Std. Error"],
        Q0 = coef(nlsLM(foodr_lhs ~ lhs(Q_0) * (exp((-(a)/lhs(Q_0)) *
            (Q_0) * foodfr)), start = list(a = 0.0001, Q_0 = 50),
            control = nls.lm.control(maxiter = 1024)))[2], Q0_se = summary(nlsLM(foodr_l
            lhs(Q_0) * (exp((-(a)/lhs(Q_0)) * (Q_0) * foodfr)), start = list(a = 0.0001,
            Q_0 = 50), control = nls.lm.control(maxiter = 1024)))$coef["Q_0",
            "Std. Error"]) |>
    slice(n()) |>
```

```
    select(subj, cond, alpha, alpha_se, Q0, Q0_se)

## # A tibble: 2 x 6
## # Groups:   subj [2]
##    subj  cond     alpha  alpha_se    Q0 Q0_se
##   <int> <int>     <dbl>     <dbl> <dbl> <dbl>
## 1     2     3 0.0000592 0.0000136  160.  23.5
## 2     3     3 0.000135  0.0000105  196.  19.5

# Social own price demand
group_by(subset(Behav, cond == 3), subj) |>
    mutate(alpha = coef(nlsLM(socr_lhs ~ lhs(Q_0) * (exp((-(a)/lhs(Q_0)) *
        (Q_0) * socfr)), start = list(a = 0.0001, Q_0 = 50), control = nls.lm.control(ma
        alpha_se = summary(nlsLM(socr_lhs ~ lhs(Q_0) * (exp((-(a)/lhs(Q_0)) *
            (Q_0) * socfr)), start = list(a = 0.0001, Q_0 = 50), control = nls.lm.contro
            "Std. Error"], Q0 = coef(nlsLM(socr_lhs ~ lhs(Q_0) * (exp((-(a)/lhs(Q_0)) *
            (Q_0) * socfr)), start = list(a = 0.0001, Q_0 = 50), control = nls.lm.contro
        Q0_se = summary(nlsLM(socr_lhs ~ lhs(Q_0) * (exp((-(a)/lhs(Q_0)) *
            (Q_0) * socfr)), start = list(a = 0.0001, Q_0 = 50), control = nls.lm.contro
            "Std. Error"]) |>
    slice(n()) |>
    select(subj, cond, alpha, alpha_se, Q0, Q0_se)

## # A tibble: 2 x 6
## # Groups:   subj [2]
##    subj  cond   alpha alpha_se    Q0 Q0_se
##   <int> <int>   <dbl>    <dbl> <dbl> <dbl>
## 1     2     3 0.00145 0.000220  64.0  15.7
## 2     3     3 0.00477 0.00117   55.5  27.7
```

2.4 Condition 4: Food own price demand

```
group_by(subset(Behav, cond == 4), subj) |>
    mutate(alpha = coef(nlsLM(foodr_lhs ~ lhs(Q_0) * (exp((-(a)/lhs(Q_0)) *
        (Q_0) * foodfr)), start = list(a = 0.0001, Q_0 = 50), control = nls.lm.control(m
        alpha_se = summary(nlsLM(foodr_lhs ~ lhs(Q_0) * (exp((-(a)/lhs(Q_0)) *
            (Q_0) * foodfr)), start = list(a = 0.0001, Q_0 = 50),
            control = nls.lm.control(maxiter = 1024)))$coef["a", "Std. Error"],
```

```
        Q0 = coef(nlsLM(foodr_lhs ~ lhs(Q_0) * (exp((-(a)/lhs(Q_0)) *
            (Q_0) * foodfr)), start = list(a = 0.0001, Q_0 = 50),
            control = nls.lm.control(maxiter = 1024)))[2], Q0_se = summary(nlsLM(foodr_l
            lhs(Q_0) * (exp((-(a)/lhs(Q_0)) * (Q_0) * foodfr)), start = list(a = 0.0001,
            Q_0 = 50), control = nls.lm.control(maxiter = 1024)))$coef["Q_0",
            "Std. Error"]) |>
    slice(n()) |>
    select(subj, cond, alpha, alpha_se, Q0, Q0_se)

## # A tibble: 2 x 6
## # Groups:   subj [2]
##    subj  cond     alpha   alpha_se    Q0 Q0_se
##   <int> <int>     <dbl>      <dbl> <dbl> <dbl>
## 1     2     4 0.0000558 0.00000469  273.  29.1
## 2     3     4 0.000131  0.00000953  193.  18.6
```

3 Cross-price Demand Curve Fitting

The following analyses were conducted to fit the demand data with the simple linear regression model and the exponential cross-price elasticity model (Hursh, 2014). It should be noted that the values of the parameters may not be exactly the same as the ones reported on the published article because of the different software algorithms were used.

3.1 Condition 1: Social cross price demand

```
# Simple linear model
group_by(subset(Behav, cond == 1), subj) |>
    mutate(Intercept = coef(lm(socr ~ 1 + foodfr))[1], Intercept_se = summary(lm(socr ~
        1 + foodfr))$coef["(Intercept)", "Std. Error"], FoodPrice = coef(lm(socr ~
        1 + foodfr))[1], FoodPrice_se = summary(lm(socr ~ 1 + foodfr))$coef["foodfr",
        "Std. Error"]) |>
    slice(n()) |>
    select(subj, cond, Intercept, Intercept_se, FoodPrice, FoodPrice_se)

## # A tibble: 4 x 6
## # Groups:   subj [4]
```

```
##    subj  cond Intercept Intercept_se FoodPrice FoodPrice_se
##   <int> <int>     <dbl>        <dbl>     <dbl>        <dbl>
## 1     1     1      25.1         4.03      25.1       0.0409
## 2     2     1      17.9         2.52      17.9       0.0255
## 3     3     1      33.8         5.57      33.8       0.0298
## 4     4     1      16.5         2.78      16.5       0.0282
```

```
# Exponential cross-price elasticity model
group_by(subset(Behav, cond == 1), subj) |>
    mutate(beta = coef(nlsLM(socr ~ log10(Q_a) + I * (exp(-b * foodfr)),
        start = list(b = 0.01, Q_a = 100, I = 1), control = nls.lm.control(maxiter = 102
        beta_se = summary(nlsLM(socr ~ log10(Q_a) + I * (exp(-b *
            foodfr)), start = list(b = 0.01, Q_a = 100, I = 1), control = nls.lm.control
            "Std. Error"], Qalone = coef(nlsLM(socr ~ log10(Q_a) +
            I * (exp(-b * foodfr)), start = list(b = 0.01, Q_a = 100,
            I = 1), control = nls.lm.control(maxiter = 1024)))[2],
        Qalone_se = summary(nlsLM(socr ~ log10(Q_a) + I * (exp(-b *
            foodfr)), start = list(b = 0.01, Q_a = 100, I = 1), control = nls.lm.control
            "Std. Error"], I = coef(nlsLM(socr ~ log10(Q_a) + I *
            (exp(-b * foodfr)), start = list(b = 0.01, Q_a = 100,
            I = 1), control = nls.lm.control(maxiter = 1024)))[3],
        I_se = summary(nlsLM(socr ~ log10(Q_a) + I * (exp(-b * foodfr)),
            start = list(b = 0.01, Q_a = 100, I = 1), control = nls.lm.control(maxiter =
            "Std. Error"]) |>
    slice(n()) |>
    select(subj, cond, beta, beta_se, Qalone, Qalone_se, I, I_se)
```

```
## # A tibble: 4 x 8
## # Groups:   subj [4]
##    subj  cond     beta beta_se   Qalone Qalone_se     I  I_se
##   <int> <int>    <dbl>   <dbl>    <dbl>     <dbl> <dbl> <dbl>
## 1     1     1 -0.00323 0.00791 9.79e-13  3.22e-10  38.9 139.
## 2     2     1 -0.00359 0.00524 5.31e-14  9.20e-12  32.9  73.2
## 3     3     1 -0.00139 0.00530 6.57e-15  4.05e-12  49.8 264.
## 4     4     1 -0.00320 0.00690 4.05e-14  9.36e-12  31.2  98.2
```

3.2 Condition 2: Food cross price demand

```r
# Simple linear model
group_by(subset(Behav, cond == 2), subj) |>
    mutate(Intercept = coef(lm(foodr ~ 1 + socfr))[1], Intercept_se = summary(lm(foodr ~
        1 + socfr))$coef["(Intercept)", "Std. Error"], FoodPrice = coef(lm(foodr ~
        1 + socfr))[1], FoodPrice_se = summary(lm(foodr ~ 1 + socfr))$coef["socfr",
        "Std. Error"]) |>
    slice(n()) |>
    select(subj, cond, Intercept, Intercept_se, FoodPrice, FoodPrice_se)
```

```
## # A tibble: 4 x 6
## # Groups:   subj [4]
##    subj  cond Intercept Intercept_se FoodPrice FoodPrice_se
##   <int> <int>     <dbl>        <dbl>     <dbl>        <dbl>
## 1     1     2      173.         8.87      173.         1.07
## 2     2     2      204.        29.9       204.         1.98
## 3     3     2      193.        20.8       193.         2.52
## 4     4     2      184.        21.0       184.         2.54
```

```r
# Exponential cross-price elasticity model
group_by(subset(Behav, cond == 2), subj) |>
    mutate(beta = coef(nlsLM(foodr ~ log10(Q_a) + I * (exp(-b * socfr)),
        start = list(b = 0.01, Q_a = 100, I = 1), control = nls.lm.control(maxiter = 102
        beta_se = summary(nlsLM(foodr ~ log10(Q_a) + I * (exp(-b *
            socfr)), start = list(b = 0.01, Q_a = 100, I = 1), control = nls.lm.control(
            "Std. Error"], Qalone = coef(nlsLM(foodr ~ log10(Q_a) +
            I * (exp(-b * socfr)), start = list(b = 0.01, Q_a = 100,
            I = 1), control = nls.lm.control(maxiter = 1024)))[2],
        Qalone_se = summary(nlsLM(foodr ~ log10(Q_a) + I * (exp(-b *
            socfr)), start = list(b = 0.01, Q_a = 100, I = 1), control = nls.lm.control(
            "Std. Error"], I = coef(nlsLM(foodr ~ log10(Q_a) + I *
            (exp(-b * socfr)), start = list(b = 0.01, Q_a = 100, I = 1),
            control = nls.lm.control(maxiter = 1024)))[3], I_se = summary(nlsLM(foodr ~
            log10(Q_a) + I * (exp(-b * socfr)), start = list(b = 0.01,
            Q_a = 100, I = 1), control = nls.lm.control(maxiter = 1024)))$coef["I",
            "Std. Error"]) |>
    slice(n()) |>
    select(subj, cond, beta, beta_se, Qalone, Qalone_se, I, I_se)
```

```
## # A tibble: 4 x 8
## # Groups:   subj [4]
```

```
##    subj  cond      beta beta_se  Qalone Qalone_se     I     I_se
##   <int> <int>     <dbl>   <dbl>   <dbl>     <dbl> <dbl>    <dbl>
## 1     1     2 -0.0129     0.289 5.98e13   5.55e17  160.    4019.
## 2     2     2 -0.000325   8.95  3.42e 6   4.31e13  198. 5469745.
## 3     3     2  0.0114     0.842 4.52e17   1.22e22  175.   11696.
## 4     4     2 -0.0268     0.280 6.48e21   3.22e25  164.    2130.
```

```r
# Get system details.
S <- benchmarkme::get_sys_details()

## Loading required package:  benchmarkme

GB <- memuse::Sys.meminfo()
```

The current machine uses the following CPU: Apple M1, with 8 cores and 16.000 GiB of RAM.

```r
sessionInfo()

## R version 4.3.3 (2024-02-29)
## Platform: aarch64-apple-darwin20 (64-bit)
## Running under: macOS Sonoma 14.3
##
## Matrix products: default
## BLAS:   /Library/Frameworks/R.framework/Versions/4.3-arm64/Resources/lib/libRblas.0.d
## LAPACK: /Library/Frameworks/R.framework/Versions/4.3-arm64/Resources/lib/libRlapack.d
##
## locale:
## [1] en_US.UTF-8/en_US.UTF-8/en_US.UTF-8/C/en_US.UTF-8/en_US.UTF-8
##
## time zone: America/Chicago
## tzcode source: internal
##
## attached base packages:
## [1] stats     graphics  grDevices utils     datasets  methods
## [7] base
##
## other attached packages:
##  [1] benchmarkme_1.0.8 minpack.lm_1.2-4  ggpubr_0.6.0
##  [4] lubridate_1.9.3   forcats_1.0.0     stringr_1.5.1
##  [7] purrr_1.0.2       tidyr_1.3.1       tibble_3.2.1
## [10] tidyverse_2.0.0   ggplot2_3.5.0     dplyr_1.1.4
## [13] readr_2.1.5       here_1.0.1        knitr_1.45
##
## loaded via a namespace (and not attached):
##  [1] utf8_1.2.4            generics_0.1.3
##  [3] rstatix_0.7.2         lattice_0.22-6
```

```
##  [5] stringi_1.8.3         hms_1.1.3
##  [7] magrittr_2.0.3        evaluate_0.23
##  [9] grid_4.3.3            timechange_0.3.0
## [11] iterators_1.0.14      foreach_1.5.2
## [13] doParallel_1.0.17     rprojroot_2.0.4
## [15] Matrix_1.6-5          backports_1.4.1
## [17] formatR_1.14          httr_1.4.7
## [19] fansi_1.0.6           scales_1.3.0
## [21] codetools_0.2-19      abind_1.4-5
## [23] cli_3.6.2             rlang_1.1.3
## [25] munsell_0.5.0         withr_3.0.0
## [27] parallel_4.3.3        tools_4.3.3
## [29] memuse_4.2-3          tzdb_0.4.0
## [31] ggsignif_0.6.4        colorspace_2.1-0
## [33] broom_1.0.5           vctrs_0.6.5
## [35] R6_2.5.1              lifecycle_1.0.4
## [37] car_3.1-2             pkgconfig_2.0.3
## [39] pillar_1.9.0          gtable_0.3.4
## [41] glue_1.7.0            benchmarkmeData_1.0.4
## [43] xfun_0.42             tidyselect_1.2.1
## [45] highr_0.10            rstudioapi_0.15.0
## [47] carData_3.0-5         compiler_4.3.3
```

```
Sys.time() - how_long
```

```
## Time difference of 1.605 secs
```