Demand Curve Analysis

Nonlinear Least Squares, Mixed Effects, and Bayesian Hierarchical Models

Yen-Yi Ho

Department of Statistics, University of South Carolina

Check for updates

Article

SMMR
STATISTICAL METHODS IN MEDICAL RESEARCH

Statistical Methods in Medical Research
2018, Vol. 27(7) 2038–2049
© The Author(s) 2016
Reprints and permissions:
sagepub.co.uk/journalsPermissions.nav
DOI: 10.1177/0962280216673675
journals.sagepub.com/home/smm
SAGE

A Bayesian hierarchical model for demand curve analysis

Yen-Yi Ho,[1] Tien Nhu Vo,[2] Haitao Chu,[3] Xianghua Luo[3,4] and Chap T Le[3,4]

Abstract

Drug self-administration experiments are a frequently used approach to assessing the abuse liability and reinforcing property of a compound. It has been used to assess the abuse liabilities of various substances such as psychomotor stimulants and hallucinogens, food, nicotine, and alcohol. The demand curve generated from a self-administration study describes how demand of a drug or non-drug reinforcer varies as a function of price. With the approval of the 2009 Family Smoking Prevention and Tobacco Control Act, demand curve analysis provides crucial evidence to inform the US Food and Drug Administration's policy on tobacco regulation, because it produces several important quantitative measurements to assess the reinforcing strength of nicotine. The conventional approach popularly used to analyze the demand curve data is individual-specific non-linear least square regression. The non-linear least square approach

Today

The science

- Self-administration experiments
- The exponential demand curve
- Q_0, α, K, P_{max}, O_{max}

Three ways to fit it

- NLIN: one subject at a time (**R**)
- Mixed effects (**SAS** `PROC NLMIXED`)
- Bayesian hierarchical (**JAGS**)

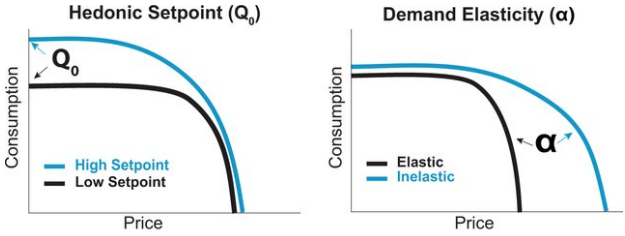
Based on: Ho, Vo, Chu, Luo, and Le (2018), *A Bayesian hierarchical model for demand curve analysis*, **Statistical Methods in Medical Research** 27(7), 2038–2049.

*This lecture is a case study in **the same model, three inferential frameworks**.*

Section 1

Part I: Motivation

Example: Demand Curves



Demand Curve Equation: $\ln Q = \ln Q_0 + k(e^{-\propto Q_0 C} - 1)$

Relative Reinforcing Efficacy

RRE describes the reinforcing potency and abuse liability of a substance.

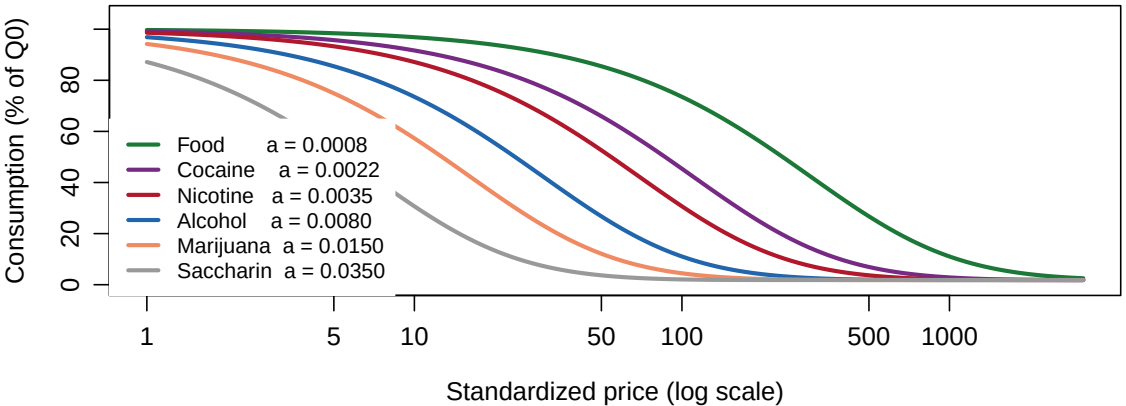
Measured through **self-administration experiments**: an animal (or a person) must "pay" for each unit of a drug.

- Psychomotor stimulants and hallucinogens
- Food
- **Nicotine**
- Alcohol

In a rodent study the **price** is the number of lever presses required per unit dose. Raise the price by cutting the unit dose.

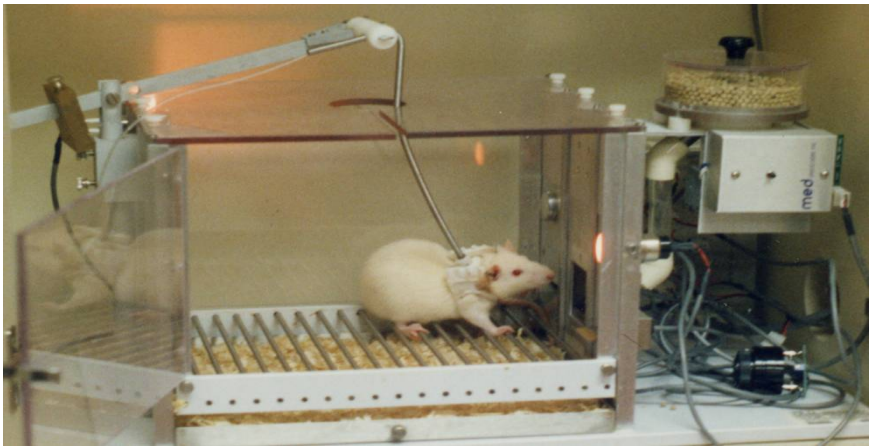
Comparing Commodities

The same model describes them all — only α changes.

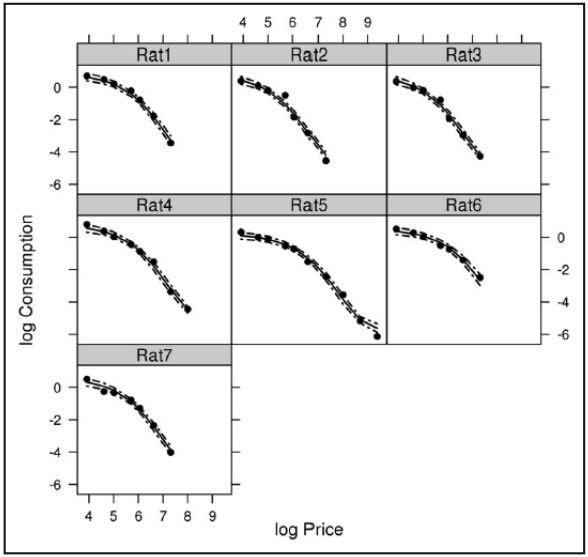


Illustrative α values on a standardized price axis, spanning the range of elasticities reported in the literature — not estimates from any one study.

A Self-Adminstration Experiment



The Data: Male



Why This Matters for Policy

The 2009 **Family Smoking Prevention and Tobacco Control Act** gives the FDA authority to regulate nicotine levels in tobacco products.

Key policy question:

*If we mandate a **reduction in nicotine content**, at what point does self-administration collapse?*

- Rodent nicotine self-administration (NSA) is the gold standard
- Human studies (cigarette purchase tasks) suffer from reporting error, non-compliance, loss to follow-up
- Demand curve analysis turns the experiment into **quantitative evidence**

The Law of Demand

Consumption generally **decreases** as price increases.

The demand curve is, conceptually, always **downward sloping**.

Two things we want to read off that curve:

1. **How much** is consumed when the drug is essentially free?
2. **How fast** does consumption fall as price rises?

The second is the one we care about: it is the measure of **reinforcing strength**.

Section 2

Part II: The Exponential Demand Model

The Model

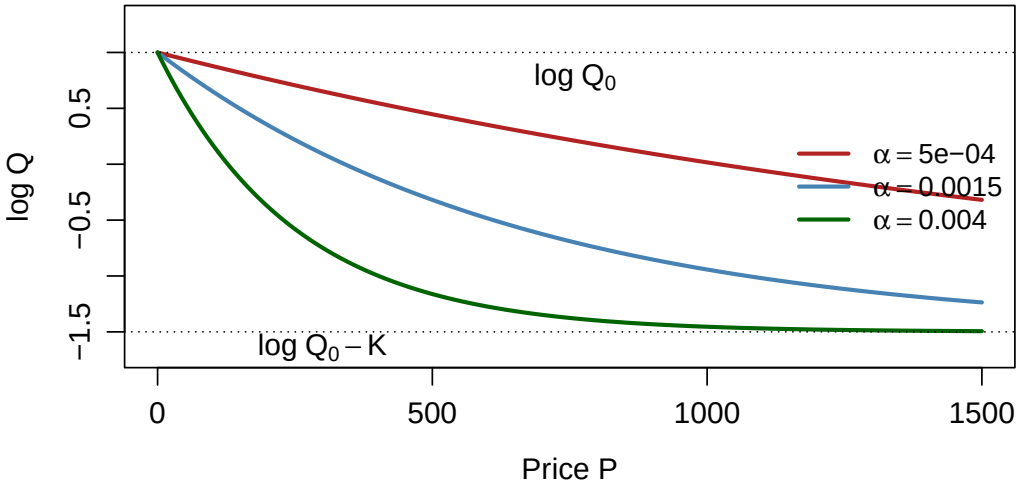
The most widely used form (Hursh and Silberberg, 2008):

$$\log Q = \log Q_0 + K\left(e^{-\alpha P} - 1\right)$$

Symbol	Meaning
Q	quantity consumed
P	unit price
Q_0	consumption as $P \to 0$ (free-access consumption)
K	span of the curve in $\log_e$ units (range of $\log Q$)
α	**elasticity** — rate of decline in relative log consumption

α is the measure of **reinforcing strength**. Small α = inelastic demand = strongly reinforcing.

Reading the Parameters Off the Curve



All three curves start at $\log Q_0$ and span K log units. Only the **rate of decline** α differs.

O_{max} and P_{max}

Two clinically relevant quantities derived from the curve.

O_{max} — the **maximum expenditure** (price $\times$ consumption) a subject is willing to spend:

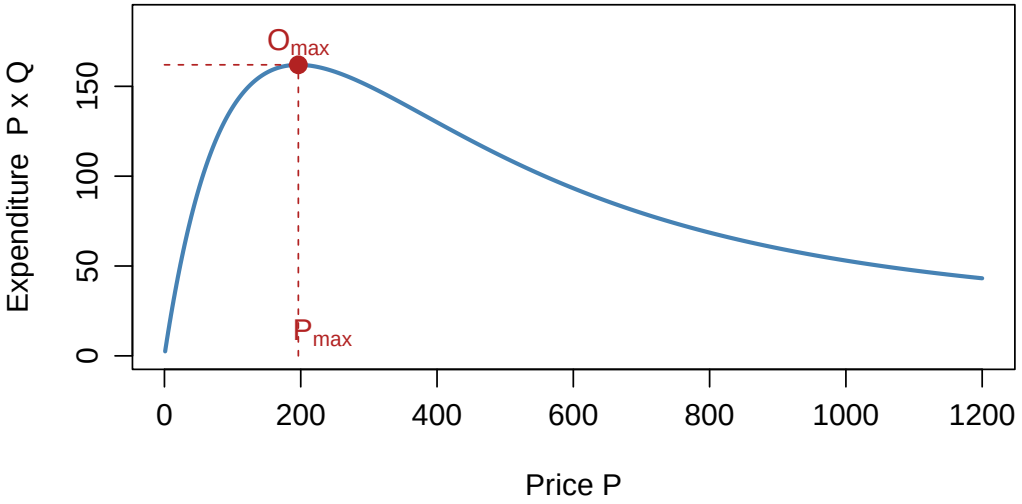
$$O_{max} = P_{max}\, Q = P_{max}\, Q_0\, e^{K(e^{-\alpha P_{max}} - 1)}$$

P_{max} — the unit price at which that maximum occurs. It has **no closed form**; it solves

$$K\,\alpha\,P_{max}\,e^{-\alpha P_{max}} = 1$$

O_{max} correlates with clinical addiction indices in both alcohol and nicotine studies.

The Expenditure Curve



Below P_{max} demand is **inelastic** (spending rises with price); above it, **elastic**.

Essential Value

Hursh (2014) converts elasticity into a single index:

$$EV = \frac{1}{100\,\alpha\,K^{1.5}}$$

Commodity	Illustrative α	EV	Reads as
Food	0.0008	1.56	strongly defended — biologically essential
Cocaine	0.0022	0.57	
Nicotine	0.0035	0.36	
Alcohol	0.0080	0.16	
Marijuana	0.0150	0.08	
Saccharin	0.0350	0.04	readily given up — a luxury good

Low $\alpha \Rightarrow$ inelastic $\Rightarrow$ high essential value.

Why α Is Hard to Estimate Well

Three practical problems with the exponential demand model:

1. K **and** α **fight each other.** The product $K\alpha$ controls the initial slope. Misfix K and α absorbs the error.
2. P_{max} **and** O_{max} **are often read off a graph.** Resolution is then limited by the sampled prices — and no standard errors come out.
3. **Subjects differ.** Each animal has its own Q_0 and its own α, and these are correlated.

The three methods in this lecture differ precisely in **how they handle these three problems**.

Section 3

Part III: The Case Study

Study Design

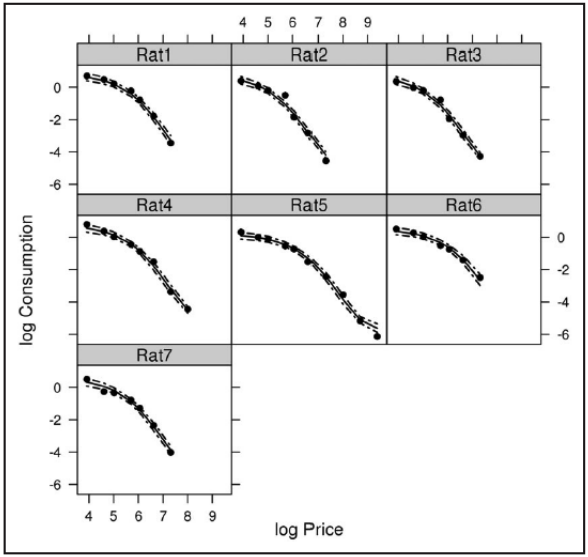
Data from Grebenstein et al. (2013), a nicotine-reduction study in rats.

- **7 male** and **7 female** Holtzman rats
- Trained to self-administer nicotine in daily **23-hour** sessions
- Price = responses required per unit dose
- Price raised **weekly** by reducing the unit dose, until responding fell to extinction levels
- Outcome: nicotine consumed (mg/kg/day), the mean of three 23-h sessions

Prices used: 50, 100, 150, 300, 429, 750, 1500, 3000, 6000, 12000.

Unbalanced: rats reach extinction at different prices, so each rat contributes a **different number** of observations (6 to 10).

The Data: Male



Section 4

Part IV: Method 1 — Nonlinear Least Squares (NLIN)

The Conventional Three-Step Approach

The method implemented in **GraphPad Prism** and used throughout the behavioral economics literature.

1. **Fix** K at the range of the observed consumption in $\log_e$ units:

$$K = \max(\log Q) - \min(\log Q)$$

   the same constant for every animal.

2. **Fit each subject separately** by minimizing the residual sum of squares for α_i and Q_{0i} — one subject at a time.

3. **Average** the individual estimates. Get 95% CIs by assuming the estimates are normally distributed.

P_{max} and O_{max} follow by plugging each subject's estimates into equations (2) and (3).

Step 1: Fix K

```
ndat <- Data1
vp   <- ndat$vP

minn <- min(ndat$logvQ)
maxx <- max(ndat$logvQ)

k <- maxx - minn          # one K for all 14 rats
```

Careful. K *is* ***ad hoc*** *— it is a function of the observed data range, not an estimated parameter. It has no standard error, and it changes if you add or drop a single extreme observation.*

Step 2: Fit Each Rat with `nls()`

```
ans <- matrix(NA, nrow = nRats, ncol = 5)
colnames(ans) <- c("alpha", "logQ0", "k", "sex", "converged")

predlogQ <- matrix(NA, nrow = nRats, ncol = length(table(Data1$sid)))

for (i in 1:nRats) {
  ans[i, 3] <- k
  wwh   <- which(ndat$sid == i)
  ans[i, 4] <- unique(ndat[wwh, 4])
  pp    <- vp[wwh]
  logvq <- ndat$logvQ[wwh]

  mynls <- try(nls(logvq ~ logQ0 + k * (exp(-alpha * pp / 10000) - 1),
                   start     = list(alpha = 10, logQ0 = 1),
                   trace     = FALSE,
                   control   = list(maxiter = 5000),
                   algorithm = "port"))
  ...
}
```

Step 2, continued: Harvesting the Fit

```
if (!inherits(mynls, "try-error")) {
  ans[i, 5] <- 1                    # 1 = converged, NA = failed
  summ <- summary(mynls)
  ans[i, 1] <- summ$coefficients[1, 1]       # alpha_i
  ans[i, 2] <- summ$coefficients[2, 1]       # logQ0_i
  predlogQ[i, 1:length(logvq)] <- logvq + summ$residuals
}
```

Three details worth pausing on:

- `try()` — with 6 to 10 points per rat, **some fits will not converge**
- `algorithm = "port"` — more robust than the default Gauss-Newton here
- `pp / 10000` — the scaling; `alpha = 10` as a start value is really 10×10^{-4}

Step 3: Average and Build Intervals

```
whmale <- which(ans[, 4] == 1)

C0     <- ans[-whmale, 2]       # female logQ0
CM     <- ans[ whmale, 2]       # male   logQ0
alpha0 <- ans[-whmale, 1]       # female alpha
alphaM <- ans[ whmale, 1]       # male   alpha

mean(alpha0)
sd(alpha0) / sqrt(length(alpha0))
mean(alpha0) + qt(0.975, 6) * sd(alpha0) / sqrt(length(alpha0))
mean(alpha0) - qt(0.975, 6) * sd(alpha0) / sqrt(length(alpha0))

t.test(alphaM, alpha0)          # sex difference in elasticity
```

Step 3: P_{max} by Numerical Optimization

P_{max} solves $K\alpha Pe^{-\alpha P} = 1$. Minimize the squared residual:

```
Pmax.s <- function(pmax) {
  (k * a * pmax * exp(-a * pmax) - 1)^2
}

pmax.i <- rep(0, length(alpha0))
for (i in 1:length(alpha0)) {
  parameter <- 9                  # starting value
  k <- ans[1, 3]
  a <- alpha0[i] / 10000
  pmax.i[i] <- optim(parameter, Pmax.s, method = "BFGS")$par
}

mean(pmax.i)
mean(pmax.i) + qt(0.975, 6) * sd(pmax.i) / sqrt(length(pmax.i))
```

Step 3: O_{max} by Substitution

```
o.maxf <- function(pmax, logq0, logk, alpha) {
  pmax * exp(logq0) * exp(exp(logk) * (exp(-alpha * pmax) - 1))
}

omax.i <- rep(NA, length(pmax.i))
for (i in 1:length(pmax.i)) {
  omax.i[i] <- o.maxf(pmax  = pmax.i[i],
                      logq0 = C0[i],
                      logk  = log(k),
                      alpha = alpha0[i] / 10000)
}

mean(omax.i)
mean(omax.i) + qt(0.975, 6) * sd(omax.i) / sqrt(length(omax.i))
```

What NLIN Gets Wrong

Problem	Consequence
K fixed at the data range	no SE for K; bias leaks into α
One subject at a time	between- and within-subject variance never separated
Averaging point estimates	ignores the precision of each subject's fit
Normal CIs for P_{max}, O_{max}	these are **skewed** — the CI is wrong
Small n_i per subject	some fits do not converge at all

It is, however, **fast**: 0.21 seconds for this dataset.

The deeper issue: NLIN answers "what is the average of 14 separate answers?" — not "what is the answer for this population of rats?"

Section 5

Part V: Method 2 — Mixed Effects Regression

The Model

Let y_{ij} be log consumption for rat i at price P_j:

$$y_{ij} = [C_0 + C_M I(\text{Male}_i) + u_i] + e^{\log K}\left\{e^{-[(\alpha_0+\alpha_M I(\text{Male}_i)+b_i)P_j]} - 1\right\} + \varepsilon_{ij}$$

Term	Meaning
C_0	log consumption at $P = 0$, female rats
C_M	male-female difference in log consumption at $P = 0$
α_0	rate of decline, female rats
α_M	male-female difference in rate of decline
u_i, b_i	random effects: rat-specific intercept and elasticity
ε_{ij}	residual, $N(0, \sigma^2)$

The Random Effects

$$\begin{pmatrix} u_i \\ b_i \end{pmatrix} \sim N\left(\begin{pmatrix} 0 \\ 0 \end{pmatrix}, \Sigma \right), \qquad \Sigma = \begin{bmatrix} \sigma_u^2 & \rho\sigma_u\sigma_b \\ \rho\sigma_u\sigma_b & \sigma_b^2 \end{bmatrix}$$

Three things this buys us that NLIN cannot give:

1. $\log K$ is now an **estimated parameter**, not a plug-in constant
2. Correlation within a rat across prices is modeled **through** the random effects — ε_{ij} stay independent
3. ρ tells us whether rats that consume more at free access also decline more slowly

Fitted by **maximum likelihood** with adaptive Gaussian quadrature.

SAS: Importing the Data

```
proc import datafile="DataF2.txt"
    out=mydata dbms=tab replace;
    getnames=yes;
    DATAROW=2;
run;

proc print data=mydata;
run;
```

The tab-delimited file has the same six columns: `vP`, `vQ`, `sid`, `Male`, `logvQ`, `logvP`.

SAS: PROC NLMIXED

```
proc nlmixed data=mydata maxiter=1000 Qpoints=100 fd;

  parms logQ0=1 alpha=10 DlogQ0=0 Dalpha=0
        logsig=-1 logsiga=0 logsigQ0=0 fz=0 logK=2;

  predlogvQ = (logQ0 + DlogQ0*Male + logQ0i)
            + exp(logK)*(exp(-(alpha + Dalpha*Male + alphai)*vP/10000) - 1);

  rho = (exp(2*fz) - 1)/(exp(2*fz) + 1);

  model logvQ ~ normal(predlogvQ, exp(2*logsig));
```

`Qpoints=100` — quadrature points. `fd` — finite-difference derivatives.

SAS: The Random Statement

```
random alphai logQ0i ~ normal([0,0],
          [exp(logsiga*2),
           rho*exp(logsiga + logsigQ0),
           exp(logsigQ0*2)])
       subject=sid OUT=alphalogQ0i;
```

SAS wants the 2×2 covariance matrix as a **lower triangle**, in order: σ_a^2, $\rho\sigma_a\sigma_{Q_0}$, $\sigma_{Q_0}^2$.

Note the **three reparameterizations** that keep the optimizer unconstrained:

Written as	Guarantees
`exp(2*logsig)`	$\sigma^2 > 0$
`exp(logsiga*2)`	$\sigma_a^2 > 0$
`rho = (e^{2z}-1)/(e^{2z}+1)`	Fisher's z: $-1 < \rho < 1$

SAS: Predictions and Back-Transformed Estimates

```
  predict predlogvQ                    out=predlogvQ;
  predict logQ0 + DlogQ0*Male + logQ0i out=plogQ0i;
  predict alpha + Dalpha*Male + alphai out=palphai;

  estimate "logQ0"  logQ0;
  estimate "DlogQ0" DlogQ0;
  estimate "alpha"  alpha;
  estimate "Dalpha" Dalpha;
  estimate "K"      exp(logK);
  estimate "siga"   exp(logsiga);
  estimate "sigQ0"  exp(logsigQ0);
  estimate "rho"    rho;
  estimate "sig"    exp(logsig);

  ods output ParameterEstimates=ParameterEstMLE
             FitStatistics=FitStatMLE
             AdditionalEstimates=AdditionalEstMLE;
run;
```

SAS: Why ESTIMATE Rather Than Reading parms

The parameters SAS optimizes are `logK`, `logsig`, `logsiga`, `fz` — none of which is the quantity we want to report.

`ESTIMATE` applies the **delta method**: it back-transforms the estimate *and* propagates the standard error.

```
proc print data=FitStatMLE; run;

data param; set ParameterEstMLE; run;

PROC EXPORT DATA=work.param
     OUTFILE="RealDataSASout-Param.txt"
     DBMS=TAB REPLACE;
RUN;
```

Exporting to tab-delimited text lets R pick the results back up for plotting and for the bootstrap.

Mixed Effects Results

For **female** rats at price near 0, consumption is $e^{0.906} = 2.48$ mg/kg — 95% CI $(1.93, 3.18)$.

- $C_M = -0.25$, 95% CI $(-0.60, 0.10)$ — males consume **less** in log scale, but **not significantly**
- $\alpha_0 = 1.053 \times 10^{-3}$, 95% CI $(6.58 \times 10^{-4},\ 1.45 \times 10^{-3})$ — significantly > 0
- $\alpha_M = -1.93 \times 10^{-4}$, 95% CI $(-7.34 \times 10^{-4},\ 3.48 \times 10^{-4})$ — **no sex difference** in reinforcing efficacy
- Overall $R^2 = 0.9865$

The Bootstrap Problem

The mixed model gives us $\log K$, α_0, α_M with standard errors.

But P_{max} and O_{max} are **nonlinear functions** of those estimates, and their sampling distributions are **skewed**.

So: resample with replacement, refit, and build a **bias-corrected and accelerated (BCa)** interval from the bootstrap histogram.

> ***Careful.*** *This is expensive. One iteration takes about* ***4.4 seconds*** *on a 2.8 GHz Xeon; 1000 iterations took* ***73 minutes****.*

The Bayesian approach in Part VI gets these intervals for free.

Section 6

Part VI: Method 3 — Bayesian Hierarchical Model

The Model

Same mean structure, now written as a conditional mean:

$$\eta_{ij} = (C_0 + C_M \text{Male}_i + u_i) + e^{\log K}\left\{e^{-[(\alpha_0+\alpha_M \text{Male}_i+b_i)P_j]} - 1\right\}$$

with $y_{ij} \sim N(\eta_{ij}, \sigma^2)$, $\sigma^2 = 1/\tau$.

Random effects:

$$\begin{pmatrix} u_i \\ b_i \end{pmatrix} \sim MVN(\mathbf{0}, \Sigma), \qquad \Sigma^{-1} \sim \text{Wishart}(\Omega, p)$$

Likelihood:

$$L(Y|\theta) = \prod_{i=1}^{N}\prod_{j=1}^{M} f(y_{ij}|\eta_{ij}, \tau)$$

Prior Specification

Non-informative, or as close as we can reasonably get:

Parameter	Prior
τ	Gamma$(10^{-3}, 10^{-3})$
C_0	$N(0, 10^3)$
C_M	$N(0, 10^3)$
α_0	$N(0, 10^3)$
α_M	$N(0, 10^3)$
$\log K$	$N(0, 10^3)$
Σ^{-1}	Wishart$\left(\begin{bmatrix} 0.1 & 0.005 \\ 0.005 & 0.1 \end{bmatrix}, 3 \right)$

Careful. *In BUGS/JAGS,* `dnorm(0, 0.001)` *is mean 0 and* ***precision*** *0.001 — i.e. variance 1000. Not standard deviation.*

The JAGS Model

```
model_string <- "
model{
  for(i in 1:tN) {
    # vP has been divided by 10000 in the R data step
    # Re[, 1] = logQ0i   Re[, 2] = alphai
    predlogvQ[i] <- (logQ0 + DlogQ0*Male[i] + Re[sid[i], 1]) +
      exp(logK) * (exp(-(alpha + Dalpha*Male[i] + Re[sid[i], 2]) * vP[i]) - 1)

    logvQ[i] ~ dnorm(predlogvQ[i], tau)      # this is the data
  }
  ...
```

Note `Re[sid[i], 1]` — JAGS looks up the random effect by **subject index**. This is why `sid` had to be renumbered $1, \ldots, 14$.

The JAGS Model, continued

```
  for(j in 1:nRats){
    Re[j, 1:2] ~ dmnorm(nu[1:2], ReV[1:2, 1:2])
  }

  tau ~ dgamma(0.001, 0.001)       # precision of epsilon_ij
  ReV[1:2, 1:2] ~ dwish(R[1:2, 1:2], 3)

  logQ0  ~ dnorm(0, 0.001)
  DlogQ0 ~ dnorm(0, 0.001)
  alpha  ~ dnorm(0, 0.001)
  Dalpha ~ dnorm(0, 0.001)
  logK   ~ dnorm(0, 0.001)

  var.tau <- 1/tau
}
"
```

`dmnorm` and `dwish` are both parameterized by the **precision** matrix. `var.tau` is a derived node — JAGS monitors it like any other.

Preparing the Data List

```
library(rjags)
set.seed(3257)

Data1 <- read.csv("DataF.csv", header = TRUE)
Data1 <- Data1[Data1$sid != 8 & Data1$sid != 16, ]
Data1$sid <- ifelse(Data1$sid > 8, Data1$sid - 1, Data1$sid)
Data1 <- Data1[, 1:6]

data <- Data1
R <- matrix(0.005, ncol = 2, nrow = 2) +
     diag(0.1 - 0.005, ncol = 2, nrow = 2)

data_list <- list(sid   = data$sid,
                  logvQ = data$logvQ,
                  vP    = data$vP / 10000,
                  nRats = length(unique(data$sid)),
                  Male  = data$Male,
                  tN    = length(data$sid),
                  nu    = c(0, 0),
                  R     = R)
```

The Scale Matrix R

```
R <- matrix(0.005, ncol = 2, nrow = 2) +
     diag(0.1 - 0.005, ncol = 2, nrow = 2)
```

This builds

$$R = \begin{bmatrix} 0.1 & 0.005 \\ 0.005 & 0.1 \end{bmatrix}$$

- Off-diagonals from `matrix(0.005, ...)`
- Diagonal topped up to 0.1 by adding $0.1 - 0.005$

Degrees of freedom 3 is the smallest value giving a proper Wishart for a 2×2 matrix — as diffuse as we can be while staying proper.

Running the Sampler

```
inits_list <- list(logQ0 = 0, DlogQ0 = 0, alpha = 10,
                   Dalpha = -2, logK = 2, tau = 20)

jags_model <- jags.model(textConnection(model_string),
                         data     = data_list,
                         inits    = inits_list,
                         n.adapt = 2000,
                         n.chains = 3,
                         quiet    = TRUE)

update(jags_model, 10000)              # burn-in

mcmc_samples <- coda.samples(jags_model,
    c("logQ0", "DlogQ0", "alpha", "Dalpha", "logK",
      "tau", "Re", "predlogvQ", "var.tau"),
    n.iter = 500000, thin = 10)
```

Why These Settings

Setting	Reason
`n.adapt = 2000`	JAGS tunes its samplers; these draws are **discarded**
`update(., 10000)`	burn-in — 10,000 iterations discarded
`n.chains = 3`	three chains are needed for the Gelman–Rubin diagnostic
`n.iter = 500000`	the posterior sample
`thin = 10`	keep every 10th draw to cut autocorrelation

Net: 5×10^4 retained draws per chain.

Starting values matter. `alpha = 10` and `logK = 2` put the chain in the right region immediately; starting at 0 can wander for a long time in a model this nonlinear.

Summarizing the Posterior

```
source("jagsresults.R")

mcmcsum <- jagsresults(mcmc_samples,
              params = c("Dalpha", "alpha", "logK",
                         "logQ0", "DlogQ0", "tau"))

meanstat <- mcmcsum[, 1]        # posterior mean
lci      <- mcmcsum[, 3]        # 2.5%  equal-tail
uci      <- mcmcsum[, 7]        # 97.5% equal-tail
```

`jagsresults()` is a small helper that pulls out only the named nodes — useful when `predlogvQ` and `Re` add 133 monitored quantities you do not want in the summary table.

Convergence Diagnostics

```
library(coda)

gelman.diag(mcmc_samples[, c("logQ0", "DlogQ0", "alpha",
                             "Dalpha", "logK", "tau")])
gelman.plot(mcmc_samples)

traceplot(mcmc_samples[, "alpha"])
autocorr.plot(mcmc_samples[[1]][, "alpha"])

effectiveSize(mcmc_samples[, "alpha"])
```

Gelman–Rubin statistic: compares between-chain to within-chain variance. All parameters here gave values close to **1**, confirming convergence.

Getting P_{max} and O_{max} for Free

This is where the Bayesian approach pays off.

```
logQ0  <- as.matrix(mcmc_samples[, "logQ0"])
DlogQ0 <- as.matrix(mcmc_samples[, "DlogQ0"])
alpha  <- as.matrix(mcmc_samples[, "alpha"])
Dalpha <- as.matrix(mcmc_samples[, "Dalpha"])
logK   <- as.matrix(mcmc_samples[, "logK"])

Pmax.s <- function(pmax) (k * a * pmax * exp(-a * pmax) - 1)^2

nsim   <- length(alpha)
pmax.i <- rep(0, nsim)
for (i in 1:nsim) {
  k <- exp(logK[i])
  a <- alpha[i] / 10000
  pmax.i[i] <- optim(9, Pmax.s)$par
}
```

O_{max} from the Posterior Draws

```
o.maxf <- function(pmax, logq0, logk, alpha) {
  pmax * exp(logq0) * exp(exp(logk) * (exp(-alpha * pmax) - 1))
}

omax.i <- o.maxf(pmax  = pmax.i,
                 logq0 = logQ0,
                 logk  = logK,
                 alpha = alpha / 10000)

mean(omax.i);  sd(omax.i)
quantile(omax.i, c(0.025, 0.975))      # equal-tail credible interval

plot(density(omax.i), main = "Posterior distribution of Omax")
```

The key idea. *Apply the transformation to* ***every draw****, and the posterior of the transformed quantity falls out. No delta method, no bootstrap, no normality assumption.*

Three Ways to Get an Interval for O_{max}

Method	How	Assumption
NLIN	t interval on 7 subject values	normality of the estimates
Mixed effects	BCa bootstrap	resampling validity
Bayesian	quantiles of the posterior draws	none beyond the model

The posterior distributions of P_{max} and O_{max} are visibly **skewed** — which is exactly why the symmetric t interval is the wrong tool.

Section 7

Part VII: Comparing the Three Fits

Parameter Estimates

Parameter	NLIN Est. (95% CI)	Mixed effects Est. (95% CI)	Bayesian Est. (95% equal-tail CI)
C_0	2.69 (1.81, 3.56)	0.91 (0.66, 1.16)	0.92 (0.69, 1.16)
C_M	0.67 (−0.25, 1.59)	−0.25 (−0.60, 0.10)	−0.28 (−0.59, 0.04)
α_0	4.40 (2.63, 6.17)	10.54 (6.58, 14.50)	10.70 (7.19, 13.92)
α_M	−0.03 (−2.00, 1.94)	−1.93 (−7.33, 3.48)	−2.46 (−6.47, 1.83)
$\log K$	2.50 (NA)	1.77 (1.73, 1.82)	1.77 (1.73, 1.82)

Estimates and SEs are multiplied by 10^4.

*Look at $\log K$: NLIN's ad hoc **2.50** versus an estimated **1.77**. That single difference propagates straight into α_0 — 4.40 versus 10.54.*

P_{max} and O_{max}

Parameter	NLIN	Mixed effects	Bayesian
P_{max} male	293.23 (151.08, 435.38)	243.13 (220.00, 364.45)	255.18 (241.72, 422.31)
P_{max} female	240.34 (116.28, 364.40)	198.70 (152.09, 292.60)	209.07 (148.02, 311.23)
O_{max} male	172.94 (126.36, 219.52)	153.73 (125.72, 267.47)	159.64 (116.74, 236.37)
O_{max} female	183.11 (124.50, 241.73)	161.66 (117.80, 216.70)	167.29 (129.08, 225.16)

All three agree on the **conclusion**: no meaningful sex difference.

They disagree on the **uncertainty** — note how much wider NLIN's P_{max} intervals are.

Goodness of Fit

R^2 = squared correlation between observed and predicted log consumption:

Method	R^2
NLIN	0.9747
Mixed effects	0.9865
Bayesian	0.9866

```
mod1 <- lm(predR ~ observed)
summary(mod1)

plot(observed, predR,
     xlab = "Observed logQ", ylab = "Predicted logQ",
     main = "Bayesian approach")
abline(mod1)
```

Cross-Validation

Two leave-out schemes, 1000 iterations each. Mean absolute error (SE):

Method	**Leave one observation out**	**Leave one rat out**
NLIN	0.254 (0.0018)	0.551 (0.0054)
Mixed effects	0.209 (0.0014)	0.478 (0.0055)
Bayesian	0.203 (0.0013)	0.485 (0.0044)

- Mixed effects and Bayesian are **comparable** to each other
- Both **outperform** NLIN under either scheme
- Leaving out a whole rat is much harder — as it should be

Section 8

Part VIII: Simulation Studies

Design

1000 datasets, half male and half female rats, 10 prices per rat.

True values:

$$C_0 = 1, \quad C_M = -0.3, \quad \alpha_0 = 10, \quad \alpha_M = -2, \quad \log K = 1.8$$

Random effects drawn from

$$\begin{pmatrix} u_i \\ b_i \end{pmatrix} \sim N\left(\begin{pmatrix} 0 \\ 0 \end{pmatrix}, \begin{bmatrix} 0.1 & 0.005 \\ 0.005 & 0.1 \end{bmatrix} \right)$$

Sample sizes: $N = 10, 20, 30, 40, 50$ rats.

Simulation Driver

```
N    <- as.numeric(commandArgs()[4])
iter <- as.numeric(commandArgs()[5])

inputf   <- paste(N, "rats1000datasets.txt", sep = "")
data.all <- read.table(inputf, header = TRUE)

run <- (iter*50 + 1):((iter + 1)*50)      # 50 datasets per job

for (simu in run) {
  data <- data.all[data.all$simu == simu, ]
  ...
  jags_model <- jags.model(textConnection(model_string),
                  data = data_list, inits = inits_list,
                  n.adapt = 2000, n.chains = 3, quiet = TRUE)
  update(jags_model, 10000)
  mcmc_samples <- coda.samples(jags_model, params,
                               n.iter = 100000, thin = 10)
}
```

commandArgs() reads job parameters from the shell — 20 array jobs of 50 datasets each.

Recording Coverage and Bias

```
icover <- matrix(NA, nrow = 1000, ncol = 6)
bias   <- matrix(NA, nrow = 1000, ncol = 6)

meanstat <- mcmcsum[, 1]
lci      <- mcmcsum[, 3]
uci      <- mcmcsum[, 7]

# alpha, true value 10
icover[simu, 1] <- (lci[3] < 10) && (10 < uci[3])
bias[simu, 1]   <- meanstat[3] - 10

# Dalpha, true value -2
icover[simu, 2] <- (lci[1] < -2) && (-2 < uci[1])
bias[simu, 2]   <- meanstat[1] - (-2)

colnames(icover) <- c("alpha","Dalpha","logK","logQ0","DlogQ0","tau")
```

Mean Relative Bias

Selected rows from the 1000-dataset simulation:

N	Parameter	NLIN	Mixed effects	Bayesian
10	α_0	-0.27	0.0061	0.0106
10	$\log K$	0.12	0.0003	-0.0002
10	C_M	0.59	-0.0126	0.0000
50	α_0	-0.46	0.0005	0.0016
50	$\log K$	0.17	0.0003	0.0002
50	C_0	-0.01	0.0035	0.0044

Careful. *The NLIN bias for α_0 gets* ***worse*** *as N grows: $-0.27 \to -0.46$. More data does not fix a misspecified K.*

Coverage of 95% Intervals

		NLIN		Mixed effects		Bayesian	
N	**Par.**	Cover	Length	Cover	Length	Cover	Length
10	α_0	0.18	3.05	0.97	2.95	0.94	2.55
10	C_0	0.74	0.76	0.88	0.57	0.93	0.66
10	$\log K$	NA	NA	0.98	0.07	0.95	0.06
50	α_0	0.00	1.47	0.95	1.14	0.94	1.10
50	C_0	0.73	0.41	0.87	0.27	0.95	0.29
50	$\log K$	NA	NA	0.95	0.03	0.96	0.03

NLIN coverage for α_0 collapses to **zero** at $N = 50$ — the interval is both biased and too narrow to reach the truth.

Robustness to Mis-specification

Simulate 100 rats under a **different** model — a log-linear form:

$$\log Q_{ij} = \log l + b \log p_j - a p_j + \varepsilon_{ij}$$

Then fit all three exponential demand models and compare predictions.

Method	MAE of predicted values
NLIN	22.128
Mixed effects	11.410
Bayesian	11.425

Mixed effects and Bayesian are roughly **twice as robust** when the model is wrong.

Computational Cost

For the 14-rat case study, on a single-core 2.8 GHz Xeon X5560:

Method	**Time**
NLIN	0.21 s
Mixed effects (+ bootstrap)	4362 s
Bayesian (MCMC)	3877 s

NLIN is four orders of magnitude faster — and that is its only real advantage.

For a dataset of 105 observations, an hour of compute to get honest uncertainty is a bargain.

Section 9

Part IX: Wrapping Up

A General Form

The mixed effects model extends to arbitrary covariates:

$$y_{ij} = \left(C_0 + \sum_{k=1}^{p} \beta_k X_k + u_i\right) + e^{\log K}\left[e^{-(\alpha_0 + \sum_{k=1}^{q} \gamma_k Z_k + b_i)P_j} - 1\right] + \varepsilon_{ij}$$

- X_k: covariates shifting **free-access consumption** (β_k)
- Z_k: covariates shifting **elasticity** (γ_k)

In our case study, $X = Z = I(\text{Male})$, giving C_M and α_M.

The Bayesian version is identical, with η_{ij} in place of y_{ij} and priors on every coefficient.

Choosing a Method

Situation	Use
Quick exploratory look, large n per subject	NLIN
Need SEs for α, C_0, $\log K$ in one model	Mixed effects
Need intervals for P_{max}, O_{max}	**Bayesian**
Small-to-moderate sample size	**Bayesian**
Skewed derived quantities	**Bayesian**
Matching published Prism output	NLIN

*The mixed effects model and the Bayesian model perform **comparably** here. The Bayesian approach wins on the derived quantities, where the bootstrap would otherwise cost you an hour.*

Programming Takeaways

Across all three

- Rescale P so parameters are order 1
- Reparameterize variances as $\exp(\cdot)$
- Reparameterize ρ via Fisher's z
- Wrap per-subject fits in `try()`
- Subject IDs must be $1, \ldots, n$

Tool-specific

- SAS: `ESTIMATE` back-transforms *and* propagates SEs
- SAS: `ODS OUTPUT` $\rightarrow$ tab file $\rightarrow$ R
- JAGS: `dnorm` takes **precision**
- JAGS: transform the **draws**, not the summaries
- JAGS: 3 chains for Gelman–Rubin

Key Takeaways

1. The **exponential demand curve** turns a self-administration experiment into interpretable parameters: Q_0, α, K, P_{max}, O_{max}.
2. **NLIN** fixes K ad hoc and fits one subject at a time. It is fast, it is the field standard, and it is **biased** — increasingly so as N grows.
3. **Mixed effects** estimates K, separates between- from within-subject variation, and needs a **bootstrap** for P_{max} and O_{max}.
4. **Bayesian hierarchical** matches the mixed model's performance and delivers the full posterior for every derived quantity **for free**.
5. The advantage of the model-based approaches is largest exactly where animal studies live: **small samples**.

References

- Ho, Y.-Y., Vo, T.N., Chu, H., Luo, X., and Le, C.T. (2018). A Bayesian hierarchical model for demand curve analysis. *Statistical Methods in Medical Research* **27**(7), 2038–2049.
- Hursh, S.R. and Silberberg, A. (2008). Economic demand and essential value. *Psychological Review* **115**, 186–198.
- Grebenstein, P., Burroughs, D., Zhang, Y., et al. (2013). Sex differences in nicotine self-administration in rats during progressive unit dose reduction. *Pharmacol Biochem Behav* **114**, 70–81.
- Hursh, S.R. (2014). Behavioral economics and the analysis of consumption and choice. In *The Wiley Blackwell Handbook of Operant and Classical Conditioning*.
- Kearns, D.N., Kim, J.S., Tunstall, B.J., and Silberberg, A. (2016). Essential values of cocaine and non-drug alternatives predict the choice between them. *Addiction Biology* **22**(6), 1501–1514.
- Plummer, M. (2003). JAGS: a program for analysis of Bayesian graphical models using Gibbs sampling. https://mcmc-jags.sourceforge.io

Questions?