A Note on Implementing the Pareto/NBD Model in MATLAB

Peter S. Fader
www.petefader.com

Bruce G. S. Hardie
www.brucehardie.com

Ka Lok Lee[†]
www.kaloklee.com

March 2005

1. Introduction

This note describes a MATLAB-based implementation of the Pareto/NBD model (Schmittlein et al. 1987). For illustrative purposes, we replicate and extend the basic analyses reported in Fader et al. (2005a,b).

There are four general aspects to the implementation of this model:

1. estimation of model parameters,
2. generation of an aggregate sales forecast given these parameter estimates,
3. computation of the probability that a particular customer is still "active", given information about his past behavior and the parameter estimates, and
4. prediction of a particular customer's future purchasing, given information about his past behavior and the parameter estimates.

The specific steps are outlined in sections 3–6 below. Section 2 briefly describes the nature of the data used for model calibration.

Please note that the code presented in this note is not supported by the authors.

No reader should attempt to use this code unless they can, for example, write their own MATLAB programs to estimate the parameters of simpler models such as the NBD and BG/NBD (Fader et al. 2005a). And any reader with such experience is clearly in a position to write their own code to implement the Pareto/NBD model ... and therefore has no need for this note.

The following code requires MATLAB 6.0 (R12) or later.

2. Data

Be it for parameter estimation or the prediction of future customer-level behavior, the Pareto/NBD requires three pieces of information about each customer's purchasing history: his "recency" (when his last transaction occurred), "frequency" (how many transactions he made in a specified time period), and the length of time over which we have observed his purchasing behavior. The notation used to represent this information is $(X = x, t_x, T)$, where x is the number of transactions observed in the time period $(0, T]$ and t_x $(0 < t_x \leq T)$ is the time of the last transaction. (If $t_x = 0$, $x = 0$.)

The `Individual-level Data` worksheet in the `cdnow_data.xls` workbook contains data for a sample of 2357 CDNOW customers who made their first-ever purchase at the web site during the first quarter of 1997. We have information on their repeat purchasing behavior up to the end of week 39 of 1997. In addition to this $(X = x, t_x, T)$ information, we also have information on the number of transactions made by each customer in the following 39-week period (where week 78 ended on 1998-06-30); this is denoted by `p2x`.

While the basic unit of time is one week, we recognize that transactions can occur on each day of the week. Consider customer 0001 (row 2); the number of days (expressed in terms of fractional weeks) during which *repeat* transactions could have occurred is $T = 38.86$, which implies this customer made his first-ever purchase at CDNOW on the first day of the first week of 1997. Over this time period, this customer made $x = 2$ repeat purchases, with the second repeat purchase occurring on the third day of the 30th week of 1997 ($t_x = 30.43$). This customer made one additional purchase in the following 39 weeks.

This dataset is read into MATLAB using the following script:

```
% load_data.m
%
% Script to load the CDNOW data from the spreadsheet cdnow_data.xls
%
% Assumes -- the spreadsheet cdnow_data.xls resides in the d:\ directory
%
% Peter S. Fader (http://petefader.com)
% Bruce G.S. Hardie (http://brucehardie.com)
```

```
% Ka Lok Lee (http://kaloklee.com)
%
% Last modified 2005-03-16

global p1x p2x tx T
tmpdata = xlsread('d:\cdnow_data','Individual-level Data','b2:e2358');
p1x = tmpdata(:,1);
tx = tmpdata(:,2);
T = tmpdata(:,3);
p2x = tmpdata(:,4);
clear tmpdata;
```

3. Parameter Estimation

The likelihood function for a randomly-chosen individual with purchase history $(X = x, t_x, T)$ is

$$L(r, \alpha, s, \beta \,|\, X = x, t_x, T) = \frac{\Gamma(r+x)\alpha^r \beta^s}{\Gamma(r)} \times \left\{ \frac{1}{(\alpha+T)^{r+x}(\beta+T)^s} + \left(\frac{s}{r+s+x}\right) \mathsf{A}_0 \right\}$$

where for $\alpha \geq \beta$

$$\mathsf{A}_0 = \frac{{}_2F_1\big(r+s+x, s+1; r+s+x+1; \frac{\alpha-\beta}{\alpha+t_x}\big)}{(\alpha+t_x)^{r+s+x}} - \frac{{}_2F_1\big(r+s+x, s+1; r+s+x+1; \frac{\alpha-\beta}{\alpha+T}\big)}{(\alpha+T)^{r+s+x}} \tag{1}$$

and for $\alpha \leq \beta$

$$\mathsf{A}_0 = \frac{{}_2F_1\big(r+s+x, r+x; r+s+x+1; \frac{\beta-\alpha}{\beta+t_x}\big)}{(\beta+t_x)^{r+s+x}} - \frac{{}_2F_1\big(r+s+x, r+x; r+s+x+1; \frac{\beta-\alpha}{\beta+T}\big)}{(\beta+T)^{r+s+x}}\,, \tag{2}$$

where ${}_2F_1(\cdot)$ is the Gaussian hypergeometric function.[1]

The four Pareto/NBD model parameters (r, α, s, β) can be estimated via the method of maximum likelihood in the following manner. Suppose we have a sample of N customers, where customer i had $X_i = x_i$ transactions in the period $(0, T_i]$, with the last transaction occurring at t_{x_i}. The sample log-likelihood function is given by

[1] At first glance, this expression for the Pareto/NBD likelihood function does not appear to be the same as that presented in Schmittlein et al. (1987); but rest assured, it is correct. This specific expression, along with the other results implemented in this note, is derived in Fader and Hardie (2005).

$$LL(r,\alpha,s,\beta) = \sum_{i=1}^{N} \ln\left[L(r,\alpha,s,\beta \,|\, X_i = x_i, t_{x_i}, T_i)\right].$$

This can be maximized using standard numerical optimization routines. Of course, this assumes it is easy for us to evaluate the Gaussian hypergeometric function for a given set of parameters.

The Gaussian hypergeometric function is the power series of the form

$${}_2F_1(a,b;c;z) = \sum_{j=0}^{\infty} \frac{(a)_j (b)_j}{(c)_j} \frac{z^j}{j!}, \quad c \neq 0, -1, -2, \ldots,$$

where $(a)_j$ is Pochhammer's symbol, which denotes the ascending factorial $a(a+1)\cdots(a+j-1)$. (Note that an ascending factorial can be represented as the ratio of two gamma functions, $(a)_j = \Gamma(a+j)/\Gamma(a)$.) The series converges for $|z| < 1$ and is divergent for $|z| > 1$; if $|z| = 1$, the series converges for $c - a - b > 0$.

Writing

$${}_2F_1(a,b;c;z) = \sum_{j=0}^{\infty} u_j, \text{ where } u_j = \frac{(a)_j (b)_j}{(c)_j} \frac{z^j}{j!}$$

we have the following recursive expression for each term of the series:

$$\frac{u_j}{u_{j-1}} = \frac{(a+j-1)(b+j-1)}{(c+j-1)j} z, \; j = 1, 2, 3, \ldots$$

where $u_0 = 1$.

This lends itself to a simple (and relatively robust) numerical method for the evaluation of the Gaussian hypergeometric function: continue adding terms to the series until u_j is less than "machine epsilon" (the smallest number that a specific computer recognizes as being bigger than zero). This is implemented in MATLAB using the following function:

```
function y = h2f1(a,b,c,z)
% h2f1 -- Gaussian hypergeometric function
%
% Computes the Gaussian hypergeometric function by series expansion,
% iterating to machine epsilon
%
% Syntax: h2f1(a,b,c,z) where a,b,c,z are scalars or column vectors.
%
% WARNING: this is *very* crude code
%   -- it doesn't perform basic checks such as |z| < 1 or c-a-b > 0
%      for |z| = 1
%   -- it doesn't recognize special cases such as a = c and b = c
%   -- it doesn't apply the relevant transformations when |z| is close
%      to 1 (so as to facilitate reliable convergence)
```

```
%   etc.
%
% Peter S. Fader (http://petefader.com)
% Bruce G.S. Hardie (http://brucehardie.com)
% Ka Lok Lee (http://kaloklee.com)
%
% Last modified 2005-03-16

lenz = length(z);
j = 0;
uj = ones(lenz,1);
y = uj;
lteps = 0;

while (lteps<lenz)
   lasty = y;
   j = j+1;
   uj = uj .*(a+j-1) .*(b+j-1) ./(c+j-1) .*z ./j;
   y = y + uj;
   lteps = sum(y==lasty);
end
```

The following function computes the value of the sample log-likelihood function for a given set of model parameters (contained in the vector `param`):

```
function [f,g]=pareto_nbd_ll(param)
% pareto_nbd_ll -- Pareto/NBD model log-likelihood
%
% Computes the log likelihood function for the Pareto/NBD model
%
% Syntax: pareto_nbd_ll(param) where the elements of param are r, alpha, s,
% and beta, respectively.
%
% Peter S. Fader (http://petefader.com)
% Bruce G.S. Hardie (http://brucehardie.com)
% Ka Lok Lee (http://kaloklee.com)
%
% Last modified 2005-03-16

global p1x tx T

r     = param(1);
alpha = param(2);
s     = param(3);
beta  = param(4);

maxab = max(alpha,beta);
absab = abs(alpha-beta);
param2 = s+1;
if alpha < beta
    param2 = r+p1x;
end

part1 = (alpha^r*beta^s/gamma(r))*gamma(r+p1x);
```

```
part2 = 1./((alpha+T).^(r+p1x).*(beta+T).^s);
if absab == 0
   F1=1./((maxab+tx).^(r+s+p1x));
   F2=1./((maxab+T).^(r+s+p1x));
else
   F1=h2f1(r+s+p1x,param2,r+s+p1x+1,absab./(maxab+tx))./...
       ((maxab+tx).^(r+s+p1x));
   F2=h2f1(r+s+p1x,param2,r+s+p1x+1,absab./(maxab+T))./...
       ((maxab+T).^(r+s+p1x));
end

f = -sum(log(part1.*(part2+(s./(r+s+p1x)).*(F1-F2))));
[f/1000 param]
g=[];
```

We use the `fmincon` routine, which comes as part of MATLAB's Optimization toolbox, to find the values of r, α, s, β that maximize the log-likelihood function (or more correctly, minimize $-LL$). The following script calls the routine:

```
% estimate_pareto_nbd.m
%
% Script to estimate the Pareto/NBD parameters
% *** requires the Optimization Toolbox ***
% *** assumes the script load_data.m has be run ***
%
% Peter S. Fader (http://petefader.com)
% Bruce G.S. Hardie (http://brucehardie.com)
% Ka Lok Lee (http://kaloklee.com)
%
% Last modified 2005-03-16

lb = .0001 * ones(1,4);
ub = 20 * ones(1,4);

initial = ones(1,4);

[params ll] = fmincon('pareto_nbd_ll',initial,[],[],[],[],lb,ub)
```

In contrast to models such as the NBD and the BG/NBD, we sometimes experience difficulties in finding the maximum of the log-likelihood function. Using starting values of $r = 1, \alpha = 1, s = 1, \beta = 1$, `fmincon` terminates at the following point: $r = 0.5532, \alpha = 10.5767, s = 0.6065, \beta = 11.6806, LL = -9595.0$. However, using starting values of $[2, 2, 2, 2]$, the optimization routine seems to "hang". For the moment, the best thing to do is to abort the optimization routine and restart using an alternative set of starting values. Trying out different starting values yields the following results:

Starting				Solution				
r	α	s	β	r	α	s	β	LL
0.5	1.0	0.5	1.0	0.5533	10.5777	0.6062	11.6681	−9595.0
2.0	2.0	2.0	2.0		abort			
1.5	1.0	2.0	0.5	0.5532	10.5773	0.6065	11.6771	−9595.0
0.5	0.6	0.2	0.1		abort			
0.2	0.5	0.4	0.1		abort			
0.1	0.5	0.4	3.0	0.5533	10.5773	0.6064	11.6736	−9595.0

But why is the optimization routine appearing to "hang" for certain sets of starting values? Let us consider the case of $[2, 2, 2, 2]$; we find that it is "hanging" when evaluating the parameter set $r = 0.0001, \alpha = 13.9431, s = 0.0001, \beta = 0.0001$. Since $\alpha > \beta$, we are evaluating A_0 as given in (1). In our dataset, there are 1411 observations for which $t_x = 0$; for these observations, the first Gaussian hypergeometric function is being evaluated at $z = (13.9431 - 0.0001)/13.9431 = 0.9999928$. For such a large value of z, the series in the `h2f1` function is taking a long time to converge. (This is the price we pay for using such a "crude" routine to evaluate the Gaussian hypergeometric function.)

The cause of this problem is the small lower bound we have specified for `fmincon`. If the lower bound was instead 0.1, $\alpha = 13.9431$ would map to $z = 0.9928$ for $t_x = 0$. Changing the lower bound in the `estimate_pareto_nbd.m` script to 0.1 (i.e., `lb = .1 * ones(1,4)`), we find that the optimization routine converges quickly for all of the starting values listed above. (If we change the lower bound for all parameters to 0.01, we find that convergence is *much* slower.)

4. Generating a Forecast of Aggregate Repeat Transactions

One way to assess the performance of the Pareto/NBD model is to see how well the model-based prediction of repeat purchasing by the cohort of 2357 customers tracks the actual number of repeat transactions over time.

For a randomly-chosen customer, the expected number of repeat transactions in a period of length t is given by

$$E[X(t) \,|\, r, \alpha, s, \beta] = \frac{r\beta}{\alpha(s-1)} \left[1 - \left(\frac{\beta}{\beta + t} \right)^{s-1} \right]$$

However, we are not interested in the expected number of repeat transactions for a randomly-chosen individual; rather we are interested in tracking (and forecasting) the total number of repeat transactions by the cohort of customers. In computing this cohort-level number, we need to account for the fact that different customers made their first purchase at CDNOW at different points in time during the first quarter of 1997, and consequently differ in the length of the time period during which they could have made repeat purchases. Given our recognition that transactions can occur on each

day of the week, we need to consider $7 \times 12 = 84$ different first-purchase dates.

Assuming that the first time a repeat purchase can occur is on the day after an individual's first, or "trial", purchase, total repeat transactions can be computed as follows:

$$\text{Total Repeat Transactions by } t = \sum_{s=1}^{84} \delta_{(t>\frac{s}{7})} n_s E[X(t - \tfrac{s}{7})]$$

where n_s is the number of customers who made their first purchase at CDNOW on day s of 1997 (and therefore have $t - \frac{s}{7}$ weeks within which to make repeat purchases) and $\delta_{(t>\frac{s}{7})} = 1$ if $t > \frac{s}{7}$, 0 otherwise.

To compute the expected number of total repeat transactions for each of the 39 "calibration period" weeks and each of the following 39 "forecast period" weeks, we first compute this quantity for each of the $7 \times 78 = 546$ days and then extract every 7th number to yield the corresponding weekly numbers. This is implemented in the following script, which also generates the associated tracking plots. (The actual repeat sales data, against which the model predictions are compared, are contained the `Cum. Repeat Sales` worksheet in the `cdnow_data.xls` workbook.)

```
% create_tracking_plot.m
%
% Script to compute the repeat sales for the CDNOW dataset and create the
% associated tracking plots (both cumulative and incremental)
%
% Assumes -- the parameter estimates are contained in the vector params
%         -- the individual-level customer data are residing in memory
%         -- the spreadsheet cdnow_data.xls resides in the d:\ directory
%
% Peter S. Fader (http://petefader.com)
% Bruce G.S. Hardie (http://brucehardie.com)
% Ka Lok Lee (http://kaloklee.com)
%
% Last modified 2005-03-16

r = params(1); alpha = params(2);
s = params(3); beta  = params(4);

% determine cohort size by day of trial
ns = [];
for i = 1:84
    ns(i) = sum((T == (273-i)/7));
end

% generate sales cumulative forecast
endwk = 78;
endday = endwk*7;
```

```
tmp1 = r*beta/(alpha*(s-1));
tmpcumsls1 = [];
for i = 1:endday
    tmp2 = (beta/(beta+i/7))^(s-1);
    tmpcumsls1(i) = tmp1*(1-tmp2);
end

tmpcumsls2 = zeros(84,endday);
for i = 1:84
    tmpcumsls2(i,:) = [ zeros(1,i) tmpcumsls1(1:endday-i) ];
end

cumrptsls = [];
dailysls = ns*tmpcumsls2;
for i = 1:endwk
    cumrptsls(i) = dailysls(i*7);
end

% load actual cumulative repeat sales data
actual = xlsread('d:\cdnow_data','Cum. Repeat Sales','b1:b78');

% create tracking plot of cumulative repeat sales (pred. vs actual)
plot(1:endwk,actual,'k',1:endwk,cumrptsls,'k--', [39 39],[0 5000],'k--');
xlabel('Week'); ylabel('Cum. Rpt Transactions');
legend('Actual','Pareto/NBD',4);
print -depsc 'cumrptsls.eps'

% create tracking plot of weekly repeat sales (pred. vs actual)
incrptsls = [ cumrptsls(1) diff(cumrptsls) ];
incactual = [ actual(1) diff(actual)' ];
plot(1:endwk,incactual,'k',1:endwk,incrptsls,'k--', [39 39],[0 150],'k--');
xlabel('Week'); ylabel('Weekly Rpt Transactions');
legend('Actual','Pareto/NBD',4);
print -depsc 'incrptsls.eps'
```

Looking at Figure 1, we note that the Pareto/NBD model predictions accurately track the actual (cumulative) sales trajectory in both the 39-week calibration period and the 39-week forecast period, under-forecasting by less than 2% at week 78.

In Figure 2, we report the week-by-week repeat-transaction numbers. The sales figures rise through week 12, as new customers continue to enter the cohort, but after that point it is a fixed group of 2357 eligible buyers. We see clearly that the Pareto/NBD model captures the underlying trend in repeat-buying behavior, albeit with obvious deviations because of promotional activities and the December holiday season.

5. Computing P(active)

The main result of Schmittlein et al. (1987) is an expression for the probability that a particular customer is still "active" given information about his past behavior; this can be written as

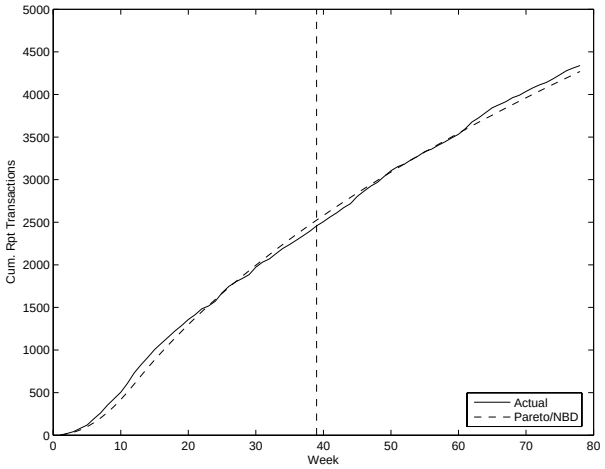

Figure 1: Tracking Cumulative Repeat Transactions

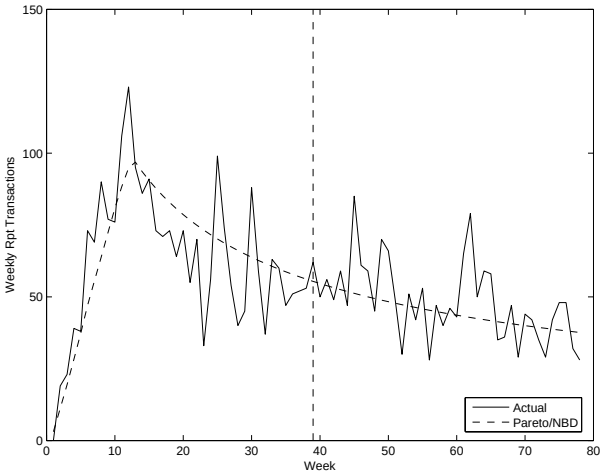

Figure 2: Tracking Weekly Repeat Transactions

$$P(\text{active} \,|\, r, \alpha, s, \beta, X = x, t_x, T) = \left\{ 1 + \left(\frac{s}{r+s+x} \right) (\alpha + T)^{r+x} (\beta + T)^s \mathsf{A}_0 \right\}^{-1}, \quad (3)$$

where A_0 is defined in (1) and (2).

The following script computes this quantity for each customer and creates a plot (to be discussed below) that helps us assess the quality of these predictions.

```
% compute_pactive.m
%
% Script to compute P(active|p1x,tx,T) for Pareto/NBD model.
% Also creates a plot comparing average P(active|p1x,tx,T) with the
% observed proportion of customers active in the second period by p1x.
%
% Assumes -- the parameter estimates are contained in the vector params
%         -- the individual-level customer data are residing in memory
%
% Peter S. Fader (http://petefader.com)
% Bruce G.S. Hardie (http://brucehardie.com)
% Ka Lok Lee (http://kaloklee.com)
%
% Last modified 2005-03-16

% compute P(active|p1x,tx,T)
r = params(1); alpha = params(2);
s = params(3); beta  = params(4);

maxab = max(alpha,beta);
absab = abs(alpha-beta);
param2 = s+1;
if alpha < beta
    param2 = r+p1x;
end

F0 = (alpha+T).^(r+p1x).*(beta+T).^s;
F1=h2f1(r+s+p1x,param2,r+s+p1x+1,absab./(maxab+tx))./...
    ((maxab+tx).^(r+s+p1x));
F2=h2f1(r+s+p1x,param2,r+s+p1x+1,absab./(maxab+T))./...
    ((maxab+T).^(r+s+p1x));
pactive = 1./(1+(s./(r+s+p1x)).*F0 .*(F1-F2));

% compute average P(active|p1x,tx,T) and determine the proportion of
% customers buying in the second 39 weeks for each level of p1x
pa_actual = zeros(max(p1x)+1,1);
pa_est = zeros(max(p1x)+1,1);
np1x = zeros(max(p1x)+1,1);
for y = unique(p1x)'
    isx = find(p1x==y);
    np1x(y+1) = length(isx);
    pa_actual(y+1) = sum(p2x(isx)>0)/np1x(y+1);
    pa_est(y+1) = sum(pactive(isx))/np1x(y+1);
end
clear y isx

% create right-censored version for plot
censor = 7;  % right-censor at 7+
denom = sum(np1x(censor+1:length(np1x)));

pa_act_cen = pa_actual(1:censor);
pa_act_cen(censor+1) = (np1x(censor+1:length(np1x))'*...
    pa_actual(censor+1:length(np1x)))/denom;
```

```
pa_est_cen = pa_est(1:censor);
pa_est_cen(censor+1) = (np1x(censor+1:length(np1x))'*...
    pa_est(censor+1:length(np1x)))/denom;

plot([0:censor],pa_act_cen,'k',[0:censor],pa_est_cen,'kp--');
legend('Empirical','Pareto/NBD',4);
xlabel('# Transactions in Weeks 1-39'); ylabel('P(active)');
axis([-.3 7.3 0 1]);
label = [ ' 0'; ' 1'; ' 2'; ' 3'; ' 4'; ' 5'; ' 6'; '7+' ];
set(gca,'xticklabel',label);
print -depsc 'pactive_grouped.eps'
```

We assess the quality of these 2357 individual-level probabilities in the following manner. For each level of repeat purchasing (x) in the first 39 weeks, we compute the average of the individual-level $P(\text{active} \,|\, X = x, t_x, T)$ numbers. We also compute the proportion of customers who were active (i.e., made at least one purchase) in the second 39-week period, for each level of repeat purchasing (x) in the first 39 weeks. These two sets of numbers are reported in Figure 3.

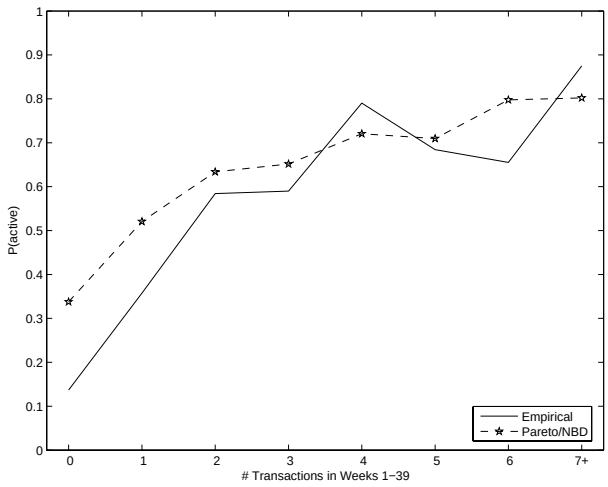

Figure 3: Predicted vs. Actual Proportions of Active Customers

The averages of the individual-level $P(\text{active} \,|\, \text{data})$ numbers should represent an upper-bound on the proportion of customers who will purchase in subsequent periods. (Note that this is not a smooth curve as, for each x, we are averaging over individuals with different values of T and t_x.) For most levels of repeat purchasing in the first 39-week period, the Pareto/NBD model provides a good estimate of the proportion of active customers.

6. Computing Conditional Expectations

Let $E(Y(t) \,|\, X = x, t_x, T)$ denote the expected number of transactions in the period $(T, T + t]$ for an individual with observed behavior $(X = x, t_x, T)$. This *conditional expectation* is given by

$$E(Y(t) \,|\, X = x, t_x, T, r, \alpha, a, b) = \frac{(r+x)(\beta+T)}{(\alpha+T)(s-1)} \left[1 - \left(\frac{\beta+T}{\beta+T+t}\right)^{s-1}\right] \times P(\text{active} \,|\, r, \alpha, s, \beta, X = x, t_x, T)\,,$$

where the expression for $P(\text{active} \,|\, \text{data})$ is given in (3).

The following script computes this quantity for each of the 2357 customers in our sample, and creates a plot that reports the average of these numbers, along with the average of the actual number of transactions that took place in the forecast period, broken down by the number of repeat purchases in the first 39 weeks.

```
% compute_ce.m
%
% Script to compute the Pareto/NBD conditional expectations
%
% Assumes -- the parameter estimates are contained in the vector params
%         -- the individual-level customer data are residing in memory
%         -- pactive (for each customer) resides in memory
%
% Peter S. Fader (http://petefader.com)
% Bruce G.S. Hardie (http://brucehardie.com)
% Ka Lok Lee (http://kaloklee.com)
%
% Last modified 2005-03-16

r = params(1); alpha = params(2);
s = params(3); beta  = params(4);

t = 39;  % period for which conditional expectations are to be computed
tmp1 = (r+p1x).*(beta+T)./((alpha+T).*(s-1));
tmp2 = ((beta+T)./(beta+T+t)).^(s-1);
ce = tmp1.*(1-tmp2).*pactive;

% compute average E[Y(t)|p1x,tx,T] and average actual number of
% transactions in the second 39 weeks for each level of p1x
ce_act = zeros(max(p1x)+1,1);
ce_est = zeros(max(p1x)+1,1);
np1x = zeros(max(p1x)+1,1);
for y = unique(p1x)'
    isx = find(p1x==y);
    np1x(y+1) = length(isx);
    ce_act(y+1) = sum(p2x(isx))/np1x(y+1);
    ce_est(y+1) = sum(ce(isx))/np1x(y+1);
```

```
end
clear y isx

% create right-censored version for plot
censor = 7;  % right-censor at 7+
denom = sum(np1x(censor+1:length(np1x)));

ce_act_cen = ce_act(1:censor);
ce_act_cen(censor+1) = (np1x(censor+1:length(np1x))'...
    *ce_act(censor+1:length(np1x)))/denom;

ce_est_cen = ce_est(1:censor);
ce_est_cen(censor+1) = (np1x(censor+1:length(np1x))'...
    *ce_est(censor+1:length(np1x)))/denom;

plot([0:censor],ce_act_cen,'k',[0:censor],ce_est_cen,'kp--');
legend('Actual','Pareto/NBD',4);
xlabel('# Transactions in Weeks 1-39');
ylabel('Average # Transactions in Weeks 40-78');
axis([-.3 7.3 0 7]);
label = [ ' 0'; ' 1'; ' 2'; ' 3'; ' 4'; ' 5'; ' 6'; '7+' ];
set(gca,'xticklabel',label);
print -depsc 'ce_plot.eps'
```

These conditional expectations are reported in Figure 4. We observe that the Pareto/NBD model provides excellent predictions of the expected number of transactions in the 39-week forecast period.

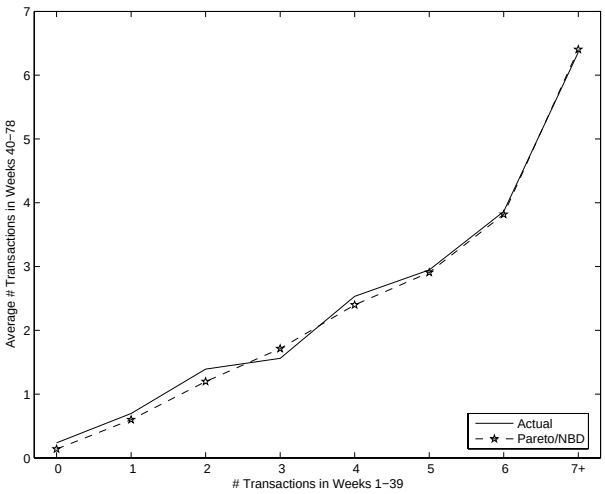

Figure 4: Conditional Expectations of Purchasing (Weeks 40–78)

References

Fader, Peter S. and Bruce G. S. Hardie (2005), "A Note on Deriving the Pareto/NBD Model and Related Expressions." <http://brucehardie.com/notes/009/>

Fader, Peter S., Bruce G. S. Hardie, and Ka Lok Lee (2005a), ""Counting Your Customers" the Easy Way: An Alternative to the Pareto/NBD Model," *Marketing Science*, forthcoming.

Fader, Peter S., Bruce G. S. Hardie, and Ka Lok Lee (2005b), "RFM and CLV: Using Iso-value Curves for Customer Base Analysis," *Journal of Marketing Research*, forthcoming.

Schmittlein, David C., Donald G. Morrison, and Richard Colombo (1987), "Counting Your Customers: Who Are They and What Will They Do Next?" *Management Science*, **33** (January), 1–24.