Implementing the S_{BB}-G/B Model in MATLAB

Peter S. Fader
www.petefader.com

Bruce G. S. Hardie[†]
www.brucehardie.com

January 2011

This note documents a somewhat informal MATLAB-based implementation of the S_{BB}-G/B model, replicating and extending the analyses reported in Section 5 of Fader, Hardie and Shang (2010). It is assumed that the reader is familiar with the concepts of the method of maximum simulated likelihood, the basics of evaluating integrals using Monte Carlo integration, the use of the Cholesky decomposition of the covariance matrix when generating draws from a multivariate normal distribution, and so on. It is also assumed that the reader is familiar with the mathematics and implementation of the BG/BB model (e.g., has worked through both Fader et al. (2010) and Fader and Hardie (2011).)

1. We load the recency/frequency summary of the annual donation behavior by the 1995 cohort of first-time supporters (Table 2 in the paper) into MATLAB using the following script:

```
% load_data.m
global x tx n num
tmp = [ 6 6 6 1203,
        5 6 6  728,
        4 6 6  512,
        3 6 6  357,
        2 6 6  234,
        1 6 6  129,
        5 5 6  335,
        4 5 6  284,
        3 5 6  225,
        2 5 6  173,
        1 5 6  119,
```

[†]© 2011 Peter S. Fader and Bruce G. S. Hardie. This document and the associated MATLAB files can be found at < http://brucehardie.com/notes/023/ >.

```
        4 4 6  240,
        3 4 6  181,
        2 4 6  155,
        1 4 6   78,
        3 3 6  322,
        2 3 6  255,
        1 3 6  129,
        2 2 6  613,
        1 2 6  277,
        1 1 6 1091,
        0 0 6 3464 ];
x   = tmp(:,1);
tx  = tmp(:,2);
n   = tmp(:,3);
num = tmp(:,4);
clear tmp
```

2. We generate two vectors of random numbers drawn from a normal distribution with mean zero and variance one using the following script. (For the purposes of this analysis, we are using 100,000 draws to evaluate the integrals of interest.)

```
% generate_common_random_numbers.m
global Z
randn('state',100); % set the seed for purposes of replication
Z = randn(100000,2);
```

3. This model sees us replacing the beta distributions for p and θ with logit-normal distributions. We will first consider the uncorrelated model.

   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]= SbbGB_ll_uncorr(param)
% SbbGB_ll_uncorr.m -- evaluate the log-likelihood
% function for the uncorrelated S_BB-G/B model
global x tx n num Z
%
% Part A
Mu = [param(1) param(2)];
Sigma = diag([param(3) param(4)]);
Y = Z*sqrt(Sigma);
logitp = Y(:,1) + Mu(1);
p = exp(logitp)./(1+exp(logitp));
logitt = Y(:,2) + Mu(2);
t = exp(logitt)./(1+exp(logitt));
%
% Part B
```

```
lik = [];
for i = 1:22
   tmp_lik = p.^x(i).*(1-p).^(n(i)-x(i)).*(1-t).^(n(i));
   for j = 0:(n(i)-tx(i)-1)
      tmp_lik = tmp_lik + p.^x(i).*(1-p).^(tx(i)...
          -x(i)+j).*t.*(1-t).^(tx(i)+j);
   end;
   ll(i) = log(mean(tmp_lik));
end;
%
f = -ll*num;
[f/1000 param]
g=[];
```

The first and second elements of `param` are the mean and variance of the normal distribution, the draws from which are transformed into values of p using the logistic transformation in "Part A". Similarly, the third and fourth elements of `param` are the mean and variance of the normal distribution whose draws are transformed into values of θ.

Now that we have, in this case, 100,000 draws from the logit-normal distributions for p and θ, "Part B" sees us performing the Monte Carlo integration: computing the likelihood function (equation (4) in the paper) for each draw and taking the average of the resulting quantity. The log of this is the value of the (simulated) log-likelihood function for the parameter values contained in `param`.

4. We use the `fmincon` routine (which is part of the Optimization Toolbox) to find the maximum of the log-likelihood function (or, more correctly, the minimum of $-$LL) via the following script:

```
% estimate_SbbGB_uncorr.m
lb = -10*[1 1 0 0];
ub =  10*[1 1 1 1];
initial = .1*ones(1,4);
[params ll] = fmincon('SbbGB_ll_uncorr',initial,[],[],[],[],lb,ub)
```

5. `fmincon` terminates at

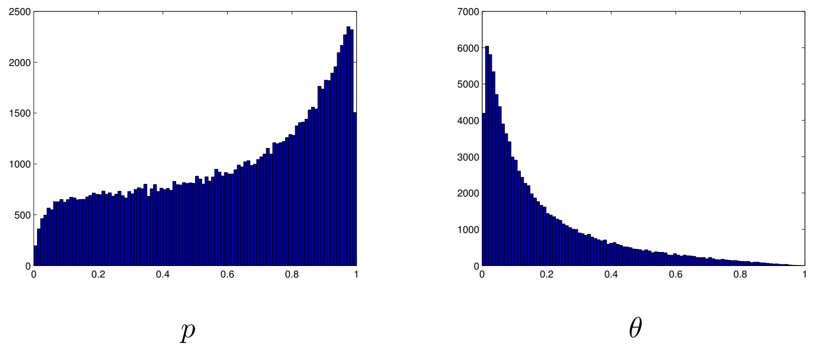
```
params = 0.7195   -1.9931    3.1782    2.2190
ll = 3.3226e+004
```

These are the "Uncorr" parameters results reported in Table 10 in the paper. The moments in (P, Θ) space are generated using the following script:

```
% uncorr_moments.m
Mu = [params(1) params(2)];
```

```
Sigma  = diag([params(3) params(4)]);
Y = Z*sqrt(Sigma);
logitp = Y(:,1) + Mu(1);
p = exp(logitp)./(1+exp(logitp));
logitt = Y(:,2) + Mu(2);
t = exp(logitt)./(1+exp(logitt));
EP = mean(p); VP = mean(p.*p)-EP^2;
ET = mean(t); VT = mean(t.*t)-ET^2;
```

6. The commands `hist(p,100)` and `hist(t,100)` give us the following plots of the distributions of P and Θ:



These are very similar to the prior plots given in Figure 7 in the paper. (Note the differences as p tends to 1 and θ tends to 0.)

7. We now run the correlated model. The only change we need to make to the log-likelihood code concerns the creation of `Y` given the two columns of uncorrelated normal random numbers contained in `Z`. Note that we are no longer directly estimating the variance of logit(P) and logit(Θ); rather we are estimating the elements of the Cholesky decomposition of the covariance matrix (`R`).

The following function computes the value of the $\mathrm{S_{BB}}$-G/B log-likelihood function:

```
function [f,g]=SbbGB_ll_corr(param)
% SbbGB_ll_corr.m -- evaluate the log-likelihood
% function for the (correlated) S_BB-G/B model
global x tx n num Z
%
% Part A
Mu = [param(1) param(2)];
R = [param(3) param(4); 0 param(5)];
Y = Z*R;
logitp = Y(:,1) + Mu(1);
p = exp(logitp)./(1+exp(logitp));
logitt = Y(:,2) + Mu(2);
t = exp(logitt)./(1+exp(logitt));
%
```

```
% Part B
lik = [];
for i = 1:22
   tmp_lik = p.^x(i).*(1-p).^(n(i)-x(i)).*(1-t).^(n(i));
   for j = 0:(n(i)-tx(i)-1)
      tmp_lik = tmp_lik + p.^x(i).*(1-p).^(tx(i)-x(i)+j).*t.*(1-t).^(tx(i)+j);
   end;
   ll(i) = log(mean(tmp_lik));
end;

f = -ll*num;
[f/1000 param]
g=[];
```

8. We find the maximum of the log-likelihood function via the following script:

```
% estimate_SbbGB_corr.m
lb = -10*ones(1,5);
ub =  10*ones(1,5);
initial = .1*ones(1,5);
[params ll] = fmincon('SbbGB_ll_corr',initial,[],[],[],[],lb,ub)
```

9. `fmincon` terminates at

```
params = 1.1148   -2.1448   -1.9663   -0.9056    1.7940
ll = 3.3211e+004
```

These are the "Corr" parameters results reported in Table 10 in the paper. The moments in (P, Θ) space are generated using the following script:

```
% corr_moments.m
Mu = [params(1) params(2)];
R  = [params(3) params(4); 0 params(5)];
Y = Z*R;
logitp = Y(:,1) + Mu(1);
p = exp(logitp)./(1+exp(logitp));
logitt = Y(:,2) + Mu(2);
t = exp(logitt)./(1+exp(logitt));
EP = mean(p); VP = mean(p.*p)-EP^2;
ET = mean(t); VT = mean(t.*t)-ET^2;
covPT = mean(p.*t)-EP*ET; corrPT = covPT/sqrt(VP*VT);
```

10. In order to compute $E[X(n)]$ — as required for creating the tracking plots — we need to take the expectation of equation (A5) over the $\mathrm{S_{BB}}$ distribution for P and Θ. This is achieved via the following code:

```
EX = [];
for s = 1:11
    EX(s) = mean(p.*(1-t)./t - p.*(1-t).^(s+1)./t);
end
```

The resulting numbers multiplied by 11,104 yield the S_{BB}-G/B numbers plotted in Figure 14a in the paper.

11. If we want to create a histogram to evaluate in-sample fit (along the lines of Figure 3 in the paper), we need to compute $P(X(n) = x)$. We do this by taking the expectation of equation (A4) over the S_{BB} distribution for P and Θ:

```
ms = 6; % time period for computation of probabilities (n)
PX = [];
for s = 0:ms
   tmp_PX = nchoosek(ms,s).*p.^s...
       .*(1-p).^(ms-s).*(1-t).^ms;
   for j = s:(ms-1)
      tmp_PX = tmp_PX + nchoosek(j,s).*p.^s.*(1-p).^(j-s).*t...
          .*(1-t).^j;
   end;
   PX(s+1) = mean(tmp_PX);
end;
```

Multiplying PX by 11,104 gives us the expected number of people making 0, 1, . . . , 6 repeat transactions between 1996 and 2001. (These numbers are not reported in the paper.)

12. In order to create the distribution of holdout-period transactions (Figure 15 in the paper), we need to compute $P(X(n, n + n^*) = x^*)$. We do this by taking the expectation of the unnumbered equation below (A6) over the S_{BB} distribution for P and Θ:

```
ms = 5; % time period for computation of probabilities (n*)
PXf = [];
for s = 0:ms
   tmp_PXf = (s==0)*(1-(1-t).^(max(n))) + nchoosek(ms,s).*p.^s...
       .*(1-p).^(ms-s).*(1-t).^(max(n)+ms);
   for j = s:(ms-1)
      tmp_PXf = tmp_PXf + nchoosek(j,s).*p.^s.*(1-p).^(j-s).*t...
          .*(1-t).^(max(n)+j);
   end;
   PXf(s+1) = mean(tmp_PXf);
end;
```

The resulting numbers multiplied by 11,104 yield the S_{BB}-G/B numbers plotted in Figure 15 in the paper.

(The file create_basic_plot_numbers.m contains the above three snippets of code.)

13. Following the logic associated with the derivation of equation (13), we obtain conditional expectation numbers for the $\mathrm{S_{BB}}$-G/B model by evaluating the following expression:

$$\begin{aligned}
&E(X(n,n+n^*)\,|\,\boldsymbol{\mu},\boldsymbol{\Sigma},x,t_x,n)\\
&\quad=\int_0^1\int_0^1\Big\{E(X(n,n+n^*)\,|\,p,\theta,\text{alive at }n)\\
&\qquad\qquad\times P(\text{alive at }n\,|\,p,\theta,x,t_x,n)\\
&\qquad\qquad\times f(p,\theta\,|\,\boldsymbol{\mu},\boldsymbol{\Sigma},x,t_x,n)\Big\}\,dp\,d\theta\\
&\quad=\frac{1}{L(\boldsymbol{\mu},\boldsymbol{\Sigma}\,|\,x,t_x,n)}\\
&\qquad\int_0^1\int_0^1\Big\{E(X(n,n+n^*)\,|\,p,\theta,\text{alive at }n)\\
&\qquad\qquad\times p^x(1-p)^{n-x}(1-\theta)^n f(p,\theta\,|\,\boldsymbol{\mu},\boldsymbol{\Sigma})\Big\}\,dp\,d\theta\,.
\end{aligned}$$

This is computed via the following script:

```
% conditional_expectations.m
m = 5; % time horizon for conditional expectation
CE  = [];
for i = 1:22
   tmp_lik = p.^x(i).*(1-p).^(n(i)-x(i)).*(1-t).^(n(i));
   tmp_CE  = (p.*(1-t)./t - p.*(1-t).^(m+1)./t).*tmp_lik ;
   for j = 0:(n(i)-tx(i)-1)
      tmp_lik = tmp_lik + p.^x(i).*(1-p).^(tx(i)-x(i)+j).*...
          t.*(1-t).^(tx(i)+j);
   end;
   lik = mean(tmp_lik);
   CE(i)  = mean(tmp_CE)/lik;
end;
```

(The same logic is used to compute other conditional quantities such as $P(\text{alive at } n+1\,|\,\boldsymbol{\mu},\boldsymbol{\Sigma},x,t_x,n)$, $P(X(n,n{+}n^*)=x^*\,|\,\boldsymbol{\mu},\boldsymbol{\Sigma},x,t_x,n)$, and $DERT(d\,|\,\boldsymbol{\mu},\boldsymbol{\Sigma},x,t_x,n)$.)

14. The following code reformats the results into "tabular" form, as presented in Table 11 in the paper:

```
TBL_CE = zeros(max(n)+1,max(tx)+1);
for i = 1:22
    TBL_CE(x(i)+1,tx(i)+1) = CE(i);
end;
```

15. In order to create the conditional expectations plots (Figure 16 in the paper), we need to average the conditional expectation numbers over

customers with different values of t_x for the “frequency” plot, and over customers with different values of x for the “recency” plot. The following code performs these calculations:

```
for i = 1:22
    tmp_tot(x(i)+1,tx(i)+1) = CE(i)*num(i);
    tmp_num(x(i)+1,tx(i)+1) = num(i);
end;
CEbyF = sum(tmp_tot,2)./sum(tmp_num,2); % as a function of frequency
CEbyR = sum(tmp_tot,1)./sum(tmp_num,1); % as a function of recency
```

References

Fader, Peter S. and Bruce G. S. Hardie (2011), “Implementing the BG/BB Model for Customer-Base Analysis in Excel.” <http://brucehardie.com/notes/010/>

Fader, Peter S., Bruce G. S. Hardie, and Jen Shang (2010), “Customer-Base Analysis in a Discrete-Time Noncontractual Setting,” *Marketing Science*, **29** (November–December), 1086–1108.