

Examples of the baseline models in SAS and MATLAB programs

Content:

- | | |
|--|----------|
| 1. Stochastic Fictitious Play (SAS) | pp. 1-4 |
| 2. Stochastic Fictitious Play (MATLAB) | pp. 4-8 |
| 3. Inertia, Sampling, and Weighting (SAS)* | pp. 8-12 |

Stochastic Fictitious Play (SFP) model - SAS

***** An example of a submission for the market entry game prediction competition ***

This example is an implementation (using SAS) of the Stochastic Fictitious Play (SFP) model described in the competition web site. This model was estimated to fit the estimation experiment.

The current program has three parts.

Part 1 reads the input (parameters of the games). It should appear without any change in all SAS submissions.

Part 2 derives the predictions of the model. The participants should express their ideas at this parts. They can use parts of the baseline models if they wish.

Part 3 creates the output file, and computes the model's scores. It should appear without any change in all SAS submissions.

In summary, the SAS submissions should be identical to the following example with the exception of Section 2.

Thus, when submitted the program should be set to read and fit the estimation experiment.

In the cover letter the submitters should state the nMSD on the estimation data set. The organizers will first run the submitted program on the estimation data set to make sure that they can replicate the states nMSD score.

Then, on Sept 2nd 2010 they will modify Part 1 and 3 of the submission (by replacing the filenames and the normalization variables) to derive the predictions of the model and its nMSD score in the competition experiment.

Note: This example can be used as a first draft of a submission. To try different models on the estimation set you should simply modify Section 2 (and download the two data sets from the competition site

```

;
*****Part 1:Input*****;
**** This part should appear without any change in all the SAS submissions;

options linesize=78 pagesize=999;
libname out ' ';

```

* This model provided the best fit among the baseline models

```

data a; infile 'ent_est_par.dat';
input problem k ph h l sf;

*****Part 2: The derivation of the prediction*****;

*****simulation of SFP model*****;

*****The parameters*****;
data a; set a;
      do lambda=1;
      do ww=.1;
output;
end; end;

data a; set a;
      nsim=200; ***number of simulations;

      **Terms used in the simulation;
      array qq{4,2} qq1-qq8; ** Propensities;
      array vv{4,2} v1-v8;   ** payoffs;
      array dec{4} dec1-dec4; **Decisions;
      array lagdec{4} ldec1-ldec4; **Previous decisons;

      **Stored statistics;
      ARRAY sent{2} sent1-sent2;   *mean entry rates in the simulation;
      ARRAY spay{2} spay1-spay2;   *mean efficiencies in the simulation;
      array salt{2} salt1-salt2;   *mean alteration rates in the simulation;

****new simulation*****;

do sim=1 to nsim;

*****Initial propensities*****;
do player=1 to 4;
  qq{player,1}=0;
  qq{player,2}=-log((.34/.66)**(1/lambda));
end; *of player;

block=1;
***** the 50 trials*****;
do t=1 to 50;
  do player=1 to 4;

    if t>1 then lagdec{player}=dec{player};

*****decisions *****;
    tend=(qq{player,2}-qq{player,1})*lambda;
    pdec1=1/(1+exp(tend));
    dec{player}=1; if ranuni(0)>pdec1 then dec{player}=2;
  end; *of player;

***statistics***;
  ent=sum(of dec1-dec4)-4;
  sent{block}=sum(0,sent{block},ent/(4*nsim*25));
  if ent>0 then spay{block}=sum(spay{block},ent*(10-k*(ent))/(4*nsim*25));

```

```

if t>1 then do player=1 to 4; altt=0;
  if dec{player} ne lagdec{player} then altt=1;
  if block=1 then salt{block} =sum(salt{block},altt/(4*nsim*24));
  if block=2 then salt{block} =sum(salt{block},altt/(4*nsim*25));
end;

**PAYOFFS**;;
state=h; if ranuni(0)>ph then state=1;
sft=sf; if ranuni(0)<.5 then sft=-sf;

do player=1 to 4;
  vv{player,1}=round(state/sft);
  if dec{player}=1 then vv{player,2}=state+10-k*(ent+1);
  if dec{player}=2 then vv{player,2}=state+10-k*(ent);

***propensities updating;
do st=1 to 2;
  qq{player,st}=(1-ww)*qq{player,st}+ww*vv{player,st};
end;

end; ***of player;

IF (T)/(25)=ROUND((T)/(25)) THEN do;
  block=block+1;
end;

end; **of trials***;
end; **of sim***;

*****Part 3: The output
*****;
***** This part should appear without any change in all the SAS
submissions*****;

data out.model_est; set a;

data c; infile 'ent_est_res.dat';
input problem k ph h l sf ent1 ent2 pay1 pay2 alt1
alt2;

data test; merge out.model_est c;
by problem k ph h l;

data test; set test;

msdent1=((ent1-sent1)**2)/0.0016490;
msdent2=((ent2-sent2)**2)/0.0014957;

msdpay1=((pay1-spay1)**2)/0.1371033;
msdpay2=((pay2-spay2)**2)/0.1188298;

msdalt1=((alt1-salt1)**2)/0.0011797;
msdalt2=((alt2-salt2)**2)/0.0014843;

msd=(msdent1 +msdent2+msdpay1+msdpay2 + msdalt1+msdalt2)/6;

```

```

proc print round; var k ph h l sf sent1 sent2 spay1 spay2 salt1 salt2 msd;

proc corr; var ent1 ent2 pay1 pay2 alt1 alt2; with sent1 sent2 spay1 spay2
salt1 salt2;

proc means; var msdent1 msdent2 msdpay1 msdpay2 msdalt1 msdalt2 msd;
run;

```

Stochastic Fictitious Play (SFP) model - MATLAB

```

%** An example of a submission for the market entry game prediction
competition ***

```

```

%This example is an implementation (using MATLAB) of the Stochastic
Fictitious Play (SFP) model described in the competition web site. This
model was estimated to fit the estimation experiment.

```

```

%The current program has three parts.

```

```

%Part 1 reads the input (parameters of the games). It should appear without
any change in all MATLAB submissions.

```

```

%Part 2 derives the predictions of the model. The participants should express
their ideas at this parts.

```

```

%They can use parts of the baseline models if they wish.

```

```

%Part 3 creates the output file, and computes the model's scores. It should
appear without any change in all
MATLAB submissions.

```

```

%In summary, the Matlab submissions should be identical to the following
example with the exception of Section 2.

```

```

%Thus, when submitted the program should be set to read and fit the
estimation experiment.

```

```

%In the cover letter the submitters should state the nMSD on the estimation
data set.

```

```

%The organizers will first run the submitted program on the estimation data
set to make sure that they can

```

```

%replicate the states nMSD score.

```

```

% Then, on Sept 2nd 2010 they will modify Part 1 and 3 of the submission
(by replacing the filenames and

```

```

%the normalization variables) to derive the predictions of the model and its
nMSD score in the

```

```

%competition experiment.

```

```

%Note: This example can be used as a first draft of a submission. To try
different models on the

```

```

%estimation set you should simply modify Section 2 (and download the two data
sets from the

```

```

%competition site

```

```

%*****Part

```

```

1:Input*****;

```

```

%**** This part should appear without any change in all the MATLAB
submissions;

```

```

load('ent_est_par.dat'); data = ent_est_par;    %to use to evaluate models
change to estpar.dat;

nt = 40;                                     %number of problems
x = data(1:nt,[1 2 3 4 5 6]);               % problem k ph h l sf

%*****PART 2: DERIVATION OF PREDICTIONS *****;
%***** simulation of SFP MODEL *****;
%*****The parameters*****;

lam=1
w=.1

nsim=200 %number of simulations;

% predictions: sent=entry rate, spay=efficiency, salt=alternation
% rate;
sent=zeros(40,2);
spay=zeros(40,2);
salt=zeros(40,2);

for sim=[1:nsim]
% loop through problems

sim=nsim

for prob = [1:nt]
    k=x(prob, 2);
    num_trials = 50;
    np=prob

    %*****initial propensities*****;
    % matrix of propensities: each column is a player and each row is;
    % an action (stay out and enter);

    q= zeros(2, 4); %propensities for each action;
    v=zeros(2,4); % payoff for each propensity/action;
    q(2,:)= -log((.34/.66)^(1/lam)) %initial propensities;

    block=1;
    dec=zeros(1,4);

    %*** the 50 trials *****;

    for t= [1:num_trials]
        trial=t
        blk=block
        if t>25
            block=2;
        end;

        %*****decisions *****;
        for player=[1:4]

```

```

    lagdec(player)=dec(player);
    tend=(q(2, player)-q(1, player))*lam;
    pdec1=1/(1+exp(tend));
    dec(player)=1;
    if rand>pdec1
        dec(player)=2;
    end
end

%*****statistics*****;
%entry rate;
ent=sum(dec)-4;
sent(prob, block)= sent(prob, block) + ent/(4*nsim*25);

% efficiency;
if ent>0
    spay(prob, block)=spay(prob, block)+ent*(10-
k*ent)/(4*nsim*25);
end;

% alternation rate;

if t>1
    for player=[1:4]
        altt=0;
        if dec(player)~=lagdec(player)
            altt=1;
        end;
        if block==1
            salt(prob, block)=salt(prob, block)+altt/(4*nsim*24);
        elseif block==2
            salt(prob, block)=salt(prob, block)+altt/(4*nsim*25) ;
        end
    end; %of player;
end

%*****payoffs *****;
ph=x(prob, 3); h=x(prob, 4); l=x(prob, 5); sf=x(prob, 6);
state=h;
if rand>ph
    state=l;
end;

sft=sf;
if rand>0.5
    sft=-sf;
end;

for player=[1:4]
    v(1, player)=round(state/sft);
    if dec(player)==1
        v(2, player)=state+10-k*(ent+1);
    elseif dec(player)==2
        v(2, player)=state+10-k*(ent);
    end;
end;

```

```

    %***updating propensities;
    for st=[1:2]
        q(st,player)=(1-w)*q(st, player)+w*v(st, player);
    end

    end; % of player;

    end ; % of t;
end; % of prob;
end; %of sim;

% ***** PART 3: THE OUTPUT *****;
%***** This part should appear without any change in all the MATLAB
submissions*****;

load('ent_est_res.dat'); data1 = ent_est_res;

res = data1(1:nt,[1 2 3 4 5 6 7 8 9 10 11 12]) % prob, k, ph, h, l, sf, ent1,
ent2, pay1, pay2, alt1, alt2;

x1=data1(1:nt, [1 2 3 4 5 6])

ent1=data1(:,7);
ent2=data1(:,8);
pay1=data1(:,9);
pay2=data1(:,10);
alt1=data1(:,11);
alt2=data1(:,12);
gent1=ent1-sent(:,1);
gent2=ent2-sent(:,2);
gpay1=pay1-spays(:,1);
gpay2=pay2-spays(:,2);
galt1=alt1-salts(:,1);
galt2=alt2-salts(:,2);

msdent1=(gent1.^2)/0.0016490;
msdent2=(gent2.^2)/0.0014957;

msdpay1=(gpay1.^2)/0.1371033;
msdpay2=(gpay2.^2)/0.1188298;

msdalt1=(galt1.^2)/0.0011797;
msdalt2=(galt2.^2)/0.0014843;

msd=(msdent1 +msdent2+msdpay1+msdpay2 + msdalt1+msdalt2)/6;

all=horzcat(x1, sent, spays, salts, msd)

```

The Inertia, Sampling, and Weighting (I-SAW) model - SAS

***** An example of a submission for the market entry game prediction competition ***

This example is an implementation (using SAS) of the Inertia sampling and Weighting (I-SAW) model described in the competition web site. This model was estimated to fit the estimation experiment.

The current program has three parts.

Part 1 reads the input (parameters of the games). It should appear without any change in all SAS submissions.

Part 2 derives the predictions of the model. The participants should express their ideas at this parts. They can use parts of the baseline models if they wish.

Part 3 creates the output file, and computes the model's scores. It should appear without any change in all SAS submissions.

In summary, the SAS submissions should be identical to the following example with the exception of Section 2.

Thus, when submitted the program should be set to read and fit the estimation experiment.

In the cover letter the submitters should state the nMSD on the estimation data set.

The organizers will first run the submitted program on the estimation data set to make sure that they can replicate the states nMSD score.

Then, on Sept 2nd 2010 they will modify Part 1 and 3 of the submission (by replacing the filenames and the normalization variables) to derive the predictions of the model and its nMSD score in the competition experiment.

Note: This example can be used as a first draft of a submission. To try different models on the estimation set you should simply modify Section 2 (and download the two data sets from the competition site

```
; *****Part 1: Input*****;
**** This part should appear without any change in all the SAS submissions;
```

```
options linesize=78 pagesize=999;
libname out '';
```

```
data a; infile 'ent_est_par.dat';
input problem k ph h l sf;
```

```
*****Part 2: The derivation of the prediction*****;
```

```
*****simulation of the I-SAW model*****;
```

```
*****The parameters*****;
```

```

data a; set a;
initial=.66;
do ro=.2;
  do wgm=.5;
    do kapa=1.5;
      do eps=.14;
        do pie=.2;

output;
end; end; end; end; end;

data a; set a;

nsim=400;

array his{4,2,50} his1-his400;
array tot{2} tot1-tot2;
array gm{4,2} gm1-gm8;

array dec{4} dec1-dec4;
array lagdec{4} ldec1-ldec4;
array lsurp{4} lsurp1-lsurp4;
array asurp{4} asurp1-asurp4;

array ipie{4} ipie1-ipie4;
array ieps{4} ieps1-ieps4;
array iro{4} iro1-iro4;
array iwgm{4} iwgm1-iwgm4;
array icas{4} icas1-icas4;

  **Stored statistics;
  ARRAY sent{2} sent1-sent2;    *mean entry rates in the simulation;
  ARRAY spay{2} spay1-spay2;    *mean efficiencies in the simulation;
  array salt{2} salt1-salt2;    *mean alteration rates in the simulation;

***new sim*****;

do ss=1 to nsim;
  do player=1 to 4;
    ipie{player}=ranuni(0)*2*pie;
    ieps{player}=ranuni(0)*2*eps;
    iro{player}=ranuni(0)*2*ro;
    iwgm{player}=ranuni(0)*2*wgmm;
    icas{player}=round(.5+ranuni(0)*2*kapa);

    do st=1 to 2;  gm{player,st}=0;
      do tt=1 to 50; his{player,st,tt}=.; end;
    end;**st;

  end; *of player;

block=1;

***** the 100 trials *****;

do t=1 to 50;

```

```

do player=1 to 4;
*****decisions *****;

lagdec{player}=dec{player};

explor=0; inertia=0; exploit=0;

*****exploration*****;
if t=1 then pexp=1;
if t>1 then pexp=ieps{player};

if ranuni(0)<pexp then do;
  explor=1;
  dec{player}=1; if ranuni(0)<initial then dec{player}=2; ** exploration;
end;

*****inertia*****;
if explor=0 then do;
  rsurp=lsurp{player}/(asurp{player}+lsurp{player});
  pinertia=ipie{player}**(rsurp);
  if ranuni(0)<pinertia then do; inertia=1; dec{player}=lagdec{player}; end;
end;

***exploit***** ;
if inertia=0 and explor=0 then do;
  exploit=1;
  ncas=icas{player};**** sample size***;
  do st=1 to 2; tot{st}=0; end;;
  do i=1 to ncas; **sampling**;
    case=round(.5+(t-1)*ranuni(0));
    if ranuni(0)<iro{player} then case=t-1;
    do st=1 to 2;
      draw=his{player,st,case};
      smem=(1-iwgm{player})*draw+iwgm{player}*gm{player,st};
      tot{st}=sum(tot{st},smem);
    end;
  end;

  dec{player}=1; if tot2>tot1 then dec{player}=2; if tot1=tot2 then
  dec{player}=1+round(ranuni(0));**choice;
end; **of exploit;
end; **of pl;

***statistics***;
ent=sum(of dec1-dec4)-4;
sent{block}=sum(0,sent{block},ent/(4*nsim*25));
if ent>0 then spay{block}=sum(spay{block},ent*(10-k*(ent))/(4*nsim*25));

if t>1 then do player=1 to 4; altt=0;
  if dec{player} ne lagdec{player} then altt=1;
  if block=1 then salt{block} =sum(salt{block},altt/(4*nsim*24));
  if block=2 then salt{block} =sum(salt{block},altt/(4*nsim*25));
end;

**PAYOFFS***;
state=h; if ranuni(0)>ph then state=1;

```

```

sft=sf; if ranuni(0)<.5 then sft=-sf;

do player=1 to 4;
  his{player,1,t}=round(state/sft);
  if dec{player}=1 then his{player,2,t}=state+10-k*(ent+1);
  if dec{player}=2 then his{player,2,t}=state+10-k*(ent);

**surprize;
if t=1 then lsurp{player}=mean(abs(his{player,2,t}-
gm{player,2}),abs(his{player,1,t}-gm{player,1}));
if t>1 then lsurp{player}=mean(abs(his{player,2,t}-
gm{player,2}),abs(his{player,1,t}-gm{player,1}),
abs(his{player,1,t}-his{player,1,t-1}),abs(his{player,2,t}-his{player,2,t-
1}));

if t=1 then do; asurp{player}=.00001; end;
if t>1 then asurp{player}=asurp{player}*(1-1/(50))+lsurp{player}*(1/(50));

***updating grand mean***;
do st=1 to 2;
  gm{player,st}=(1-1/(t))*gm{player,st}+(1/(t))*his{player,st,t};
end;

end; **of player;

IF (T)/(25)=ROUND((T)/(25)) THEN do;
  block=block+1;
end;

end; **of trials***;
end; **of sim***;

*****Part 3: The output*****;
** This part should appear without any change in all the SAS submissions**;
```

```

data out.model_est; set a;

data c; infile 'ent_est_res.dat';
input problem k ph h l sf ent1 ent2 pay1 pay2 alt1
alt2;

data test; merge out.model_est c;
by problem k ph h l;

data test; set test;

msdent1=((ent1-sent1)**2)/0.0016490;
msdent2=((ent2-sent2)**2)/0.0014957;

msdpay1=((pay1-spay1)**2)/0.1371033;
msdpay2=((pay2-spay2)**2)/0.1188298;

msdalt1=((alt1-salt1)**2)/0.0011797;

```

```
msdalt2=((alt2-salt2)**2)/0.0014843;  
msd=(msdent1 +msdent2+msdpay1+msdpay2 + msdalt1+msdalt2)/6;  
proc print round; var k ph h l sf sent1 sent2 spay1 spay2 salt1 salt2 msd;  
proc corr; var ent1 ent2 pay1 pay2 alt1 alt2; with sent1 sent2 spay1 spay2  
salt1 salt2;  
proc means; var msdent1 msdent2 msdpay1 msdpay2 msdalt1 msdalt2 msd;  
run;
```