Revealed Beliefs and the Marriage Market Return to Education: Simulation & Estimation Code ReadMe

Alison Andrew* & Abi Adams†

November 1, 2024

These files provide the code required to operationalize the "Revealed Beliefs" methodology for a T-period and D+1-option dynamic discrete choice problem. The model is formally outlined in Sections 2.2 and 2.3 of the paper. The code is structured into three main steps:

- Step 0: Set user-specific options
- Step 1: Simulate ex-ante experimental choice data for a hypothetical revealed belief experiment
- Step 2: Estimate belief parameters on the basis of the simulated ex-ante choice data

All steps are called from the file `master_script_Doptions_Tperiods`. To apply the approach to "real" experimental data, first apply Step 0 (the set-up) and then Step 2 to the user-collected data.

*University of Oxford & Institute for Fiscal Studies, alison.andrew@economics.ox.ac.uk.
†University of Oxford & Institute for Fiscal Studies, abi.adams-prassl@economics.ox.ac.uk.

Step 0: Setting User-Specific Options

0a) Set globals

Edit `step0a_set_globals` to set:

- T = Number of time periods
- D = Number of non-terminal actions (the terminal action, d=0, is always available)
- β = Discount factor
- $Nsims$ = number of simulations used in evaluating choice probabilities where no closed form solution is available (i.e. when D>1)

In the demonstration code, we set $T = 3$, $D = 2$, and $\beta = .95$.

0b) Set-up matrix of all deterministic decision states

Given T and D, `step0b_setup_states` generates a list of all deterministic decision states, $\bar{\omega}_t$, prior to the realization of idiosyncratic preference shocks ε_t and the stochastic state variable q. These are states where the terminal action has not already been chosen and there is a decision to be made. See Assumption A0.

In our demonstration code, states are defined by the time period and the complete history of the choices made in each past period. In the code, we allow for all non-terminal actions to be taken in any state and we allow for the ordering of choices to matter, i.e. taking $d_1 = 1, d_2 = 2$ results in a different state from taking $d_1 = 2, d_2 = 1$. In the worked example with $T = 3$ and $D = 2$, this results in 7 decision states that are defined by the time period and, for $t > 1$, by the history of non-terminal actions taken. Figure 1 shows the matrix of decision states generated by the code.

Figure 1: Decision States in Demonstration Code

State #	State -- t	State -- d1	State -- d2
1	1	NaN	NaN
2	2	1	NaN
3	2	2	NaN
4	3	1	1
5	3	2	1
6	3	1	2
7	3	2	2

0c) Set-up matrix of all state transitions

For each decision state (down the rows), `step0c_setup_state_transition` gives the $t+1$ state number implied by taking each non-terminal action (across the columns). Figure 2 gives the matrix the function generates for the demonstration code. For example, the first row that represents the decision made at $t = 1$. It shows that if $d_1 = 1$ is chosen, the decision maker starts period 2 in state 2. However, if $d_1 = 2$ is chosen, the decision maker starts period 2 in state 3.

Figure 2: State Transition Summary in Demonstration Code

State #	State -- t	State -- d1	State -- d2	Transition (dt=1)	Transition (dt=2)
1	1	NaN	NaN	2	3
2	2	1	NaN	4	6
3	2	2	NaN	5	7
4	3	1	1	NaN	NaN
5	3	2	1	NaN	NaN
6	3	1	2	NaN	NaN
7	3	2	2	NaN	NaN

0d) Preferences: Set-up payoff matrix for choosing terminal action

In the demonstration code, we assume that preferences are known with certainty. If preferences are estimated, this would be performed at this point.

The scripts `step0d_set_prefparams` and `step0d_utility_f` create the payoffs (net

of ε_t) associated with choosing the terminal action in each deterministic decision state, both when the realization of the stochastic state variable is low $q = L$ and high $q = H$.

`step0d_set_prefparams` sets the parameters of the utility function. `step0d_utility_f` creates the payoff matrix associated with the selecting the terminal action in each state. The demonstration code uses the following functional form for the payoff of path ending in taking the terminal action in period τ with realization of the stochastic state variable q_τ, with deterministic state variables $\bar{\omega}_\tau$:

$$\bar{U}(\tau, q_\tau, \bar{\omega}_\tau) = \mathbb{1}(q_\tau = H) + \gamma \sum_{t=1}^{\tau-1} \mathbb{1}(d_t = 1) + \kappa \sum_{t=1}^{\tau-1} \mathbb{1}(d_t = 2)$$

We set $\gamma = 0.05$ and $\kappa = -0.05$ in our worked example. Thus, in our worked example, the payoffs associated with choosing the terminal action in each state when $q = L$ and $q = H$ are as follows:

Figure 3: Payoff Matrix to Terminal Action in Demonstration Code

State #	State -- t	State -- d1	State -- d2	Payoff -- U(q=L)	Payoff -- U(q=H)
1	1	NaN	NaN	0	1
2	2	1	NaN	0.05	1.05
3	2	2	NaN	-0.05	0.95
4	3	1	1	0.1	1.1
5	3	2	1	0	1
6	3	1	2	0	1
7	3	2	2	-0.1	0.9

We also set the expected value of being in the period $T+1$ without having taken the terminal at any point (in our application, this is the expected value of still being unmarried at the end of adolescence). In the demonstration code, we set this as the value of choosing the terminal action in the last period with a low realization of the stochastic state variable plus a constant $\bar{V}$. In our worked example, we set $\bar{V} = -0.5$.

0e) Beliefs: Set "true" belief parameters (create $\pi(\bar{\omega}_t)$ and σ^2_ε)

Our key object of interest is $\pi(\bar{\omega}_t)$. This gives the subjective probability of receiving a high draw of the stochastic state variable in decision state $\bar{\omega}_t$. In the demonstration code, we allow $\pi(\bar{\omega}_t)$ to be completely unrestricted so that there are `Nstates-1` different probabilities to be estimated.[1]

[1]Note that since beliefs are identified from choice probabilities in the previous period, $\pi(\bar{\omega}_1)$ is not identified.

For demonstration purposes, we set the "true" `Nstates`$\times 1$ vector of belief parameters and the shock variance in `true_parms`. The first element of the vector corresponds to σ_ε^2 and the remaining elements correspond to $\pi(\bar{\omega}_t)$ vector (except for the first state where beliefs are not identified). The `step0e_makepi` function then converts these parameters into the `Nstates-1` belief vector.[2]

Figure 4: Probability of $q = H$ in Demonstration Code

State #	State -- t	State -- d1	State -- d2	pi -- Pr(q=H)
1	1	NaN	NaN	NaN
2	2	1	NaN	0.8
3	2	2	NaN	0.7
4	3	1	1	0.6
5	3	2	1	0.5
6	3	1	2	0.5
7	3	2	2	0.2

0f) Set options for number of respondents and number of experiments/respondent for simulations

Here, we we simply set the number of respondents (`n_resps`) and number of rounds per respondent (`n_exps`) to be simulated. Users could edit these to assess how precision for a particular example changes with sample size.

Step 1: Simulate ex-ante experimental choice data for a hypothetical revealed belief experiment

1a) Solve the dynamic problem for the "true" belief parameters

Taking preference parameters as given, we solve the model for the "true" belief parameters. For each decision state $(\bar{\omega}_t, q)$, we solve for:

- Choice probabilities: `pLow` $= Pr(d|q = L, \bar{\omega}_t)$, `pHigh` $= Pr(d|q = H, \bar{\omega}_t)$

[2] As the number of decision states increases, researchers will find it useful to put a functional form on beliefs to increase power. In this case, they should set the belief parameters and edit the `step0e_makepi` function accordingly. Beliefs can be any function of $\bar{\omega}_t$.

- Value functions: V= expected value of being in decision state $\bar{\omega}_t$ before the value of either q or ε is revealed
- Conditional value functions: $\texttt{vCond\_q} = v(\bar{\omega}_t, q, d_t)$= present discounted value (net of epsilon) of choosing d_t at t and then optimally thereafter, conditional on q

Algorithm 1 gives the steps taken to solve the dynamic problem. Note here that for the purpose of the code, the terminal action is that corresponding to $D + 1$. In the demonstration code, we calculate probabilities and evaluate expectations analytically for $D = 1$ and by simulation if $D > 1$.

Algorithm 1 Pseudocode for `step1_solve_problem_Doptions`

Input: (i) $\bar{\omega}$, vector of deterministic decision states (created at step 0b); (ii) state transition matrix (created at step 0c); (iii) preference parameters; (iv) belief & preference shock parameters.
Initialize matrices to store: (i) choice probabilities conditional on value of q, $pHigh$ and $pLow$ (dimension = $Nstates \times (D + 1)$); (ii) value function, V (dimension = $Nstates \times 1$); (iii) choice-specific conditional value functions, $vCond\_H$ and $vCond\_L$ (dimension = $Nstates \times (D + 1)$).
Calculate the value of $vCond\_H$ and $vCond\_L$ at period T. For non-terminal actions (columns 1:D), these are the continuation values $CV(\bar{\omega}_T)$. For the terminal action (column D+1), this is $\bar{U}(T, q, \omega_T)$.
for $s = Nstates : -1 : 1$ **do**
    for $d = [1 : D + 1]$ **do**
        $pHigh(s, d) = Pr\left(vCond\_H(s, d) + \varepsilon_d = \max\{vCond\_H(s, d') + \varepsilon_{d'}, \forall d'\}\right)$
        $pLow(s, d) = Pr\left(vCond\_L(s, d) + \varepsilon_d = \max\{vCond\_L(s, d') + \varepsilon_{d'}, \forall d'\}\right)$
    end for
    $V(s) = \pi(s)E(\max_{d'}\{vCond\_H(s, d') + \varepsilon_{d'}\}) + (1 - \pi(s))E(\max_{d'}\{vCond\_L(s, d') + \varepsilon_{d'}\})$
    $vCond\_H(s^{-1}(s), d^{-1}(s)) = V(s)$
    $vCond\_L(s^{-1}(s), d^{-1}(s)) = V(s)$
    ▷ **Comment:** $s^{-1}(s), d^{-1}(s)$ denote the previous period's state and action that result in the state s. Stored in state transition matrix.
end for

1b) Simulate ex-ante choice data

We randomly draw choice scenarios to simulate behavior at from the set of all decision states. In the demonstration code, we are sample all decision states woth equal weighting.[3] For each randomly sampled choice scenario, we then simulate respondent choices using the choice probabilities calculated at Step 1a.

[3]Note that this is not necessarily the most efficient way to sample decision states. Researchers can run simulated power calculations using different methods for sampling decision states.

Step 2: Estimate belief parameters on the basis of the simulated ex-ante choice data

2a) Define log likelihood function and check at starting values

The function `step2_ll_exante_Doptions` evaluates the log likelihood given data from an ex-ante choice experiment at a given set of belief parameters. For the given vector of belief parameters, the function first calculates the implied choice probabilities for each choice scenario presented to respondents, using `step1a_solve_problem_Doptions`. These choice probabilities are used to calculate the log likelihood of observing the set of data at the belief parameters.

2b) Find the set of belief parameters that maximise the log likelihood

In the demonstration code, we use MATLAB's `fminunc` to calculate the parameters that maximize the log-likelihood function. We estimate standard errors from the inverted empirically-estimated Hessian. For simulated data, researchers can compare the "true" parameters to their estimates and also explore the impact of design choices on power by assessing the standard errors. Table 5 gives the "true" parameters used in this simulation as well as estimated parameters, standard errors and confidence intervals.

Figure 5: Estimated & "True" Belief Parameters

true_parms	estimate	se	ci_lo	ci_hi
0.6	0.60231	0.0070077	0.58858	0.61605
0.8	0.81476	0.045525	0.72553	0.90399
0.7	0.72931	0.037286	0.65623	0.80239
0.6	0.59898	0.011816	0.57582	0.62214
0.5	0.50031	0.01158	0.47761	0.52301
0.5	0.50498	0.01308	0.47934	0.53062
0.2	0.199	0.015272	0.16907	0.22893

In our actual application, we use the `particleswarm` optimizer (and bootstrap the procedure to get standard errors). We found this to be more reliable at finding the global maximum with a more complex problem. We additionally show in step (2b*) that the particleswarm algorithm also works well on this worked example but is slower to converge than `fminunc`.