VFI Toolkit Workshop, pt2 - Standard Life-Cycle Models

vfitoolkit.com/2025-workshop-lse

Robert Kirkby

robertdkirkby.com

Victoria University of Wellington

VFI Toolkit

- We saw how to setup and solve basic Life-Cycle models.
- d (decision varialbe), a (endogenous state) and z (exogenous markov state).
- And basic analysis, StationaryDist, AllStats, and AgeConditionalStats.
- VFI Toolkit does much more: i.i.d shocks, panel data, human capital, portfolio-choice, conditional statistics, calibration, GMM estimation.
- Let's start our whirlwind tour!

- Rough outline for Part 2:
  1. Exogenous shocks: i.i.d, correlated shocks (e.g., VAR), semi-exogenous state.
  2. Permanent Types
  3. More on analysing model: simulate panel data, conditional stats, faster stats.

- We saw markov shocks z.
- Can also add i.i.d. exogenous shocks e.
- Beyond the basics, we set things up using *vfoptions* and *simoptions* to explain the model to VFI Toolkit.
- So we would create $vfoptions.n\_e$, $vfoptions.e\_grid$ and $vfoptions.pi\_e$, put copies in simoptions.
  pi_e will be a column vector.
- Action space: $(d, aprime, a, z, e, ...)$ [e always comes just after z]
- Explain e vars: e vars concept

See Intro to Life-Cycle Models: Life-Cycle Model 11.

Basic Life-Cycle Model 4

- Household problem

$$
\begin{aligned}
V(a,z,e,j) \quad = \quad & \max_{h,c,aprime} \frac{c^{1-\sigma}}{1-\sigma} - \psi\frac{h^{1+\eta}}{1+\eta} \\
& \quad\quad + s_j \beta E[V(aprime, zprime, eprime, j+1)|z] \\
& \text{if } j < Jr: \; c + aprime = (1+r)a + w\kappa_j h \, exp(z+e) \\
& \text{if } j >= Jr: \; c + aprime = (1+r)a + pension \\
& 0 \leq h \leq 1, aprime \geq 0 \\
& z' = \rho_z z + \epsilon', \quad \epsilon \sim N(0, \sigma_{z,\epsilon}) \\
& e \sim N(0, \sigma_e)
\end{aligned}
$$

- Let's write the code to solve this.
  Code: *WorkshopModel4.m* (and *WorkshopModel4_ReturnFn.m*)
- Will only explain which of our seven core steps we change.

1. Model action and state-spaces.
   - Exogenous i.i.d. variable: e

```
%% Model action and state-space
n_d=21;
n_a=201; % number of grid points for our endogenous
    state
n_z=9;
n_e=7;
N_j=81; % periods, represent ages 20 to 100
```

- Add $n\_e = 7$.

3. Grids

- Add *e_grid* and *pi_e* (grid and the markov transition matrix).
- To discretize a Normal distribution we use the same commands as discretize AR(1), but with correlation=0.
  The discretization process produces a markov transition matrix, we keep just the first row as i.i.d. probabilities. (Could be any row, use first row for convenience.)

```
%% Grids
% Discretize AR(1) using Farmer-Toda method
[z_grid,pi_z]=discretizeAR1_FarmerToda(0,Params.rho_z,
    Params.sigma_zepsilon,n_z);

% Discretize Normal dist using Farmer-Toda method
[e_grid,pi_e]=discretizeAR1_FarmerToda(0,0, Params.
    sigma_e,n_e);
pi_e=pi_e(1,:)'; % Just keep first row for i.i.d. (as
    column)
```

Step 2 was parameters, we need to add some, but changes are obvious.

Basic Life-Cycle Model 4

3. No longer baseline features, so have to use *vfoptions* and *simoptions* to tell VFI Toolkit.

```
vfoptions.n_e=n_e;
vfoptions.e_grid=e_grid;
vfoptions.pi_e=pi_e;
simoptions.n_e=vfoptions.n_e;
simoptions.e_grid=vfoptions.e_grid;
simoptions.pi_e=vfoptions.pi_e;
```

4. ReturnFn

```
function F=WorkshopModel2_ReturnFn(h,aprime,a,z,e,
    sigma,psi,eta,w,r,kappa_j,agej, Jr)
% first five entries are the action space

F=-Inf;

% budget constraint
if agej<Jr % working
    c=(1+r)*a +w*kappa_j*h*exp(z+e) - aprime;
else % retired
    c=(1+r)*a -aprime;
end

if c>0
    % utility fn
    F=(c^(1-sigma))/(1-sigma)-psi*(h^(1+eta))/(1+eta
        );
end
```

6. Agents stationary distribution.

```
%% Initial distribution of agents at birth (j=1)
jequaloneDist=zeros([n_a,n_z,n_e],'gpuArray'); % Put
    no households anywhere on grid
jequaloneDist(1,ceil(n_z/2),:)=shiftdim(pi_e,2); %
    start with 0 assets, median z shock, dist of e
```

- Have to say what households look like in period 1 (here, zero assets).
- Just change initial dist to have the right 'state space', which is (a, z, e)
- Mass for initial dist is 1.

Solve V and Policy unchanged. Except now the state space is (a, z, e, j), so they are different shapes.

Basic Life-Cycle Model 4

7. Generate model moments/statistics. (1/2)

```
FnsToEvaluate.earnings=@(h,aprime,a,z,e,w,kappa_j) w*kappa_j*h*exp(z+e); % w*kappa_j*h*exp(z+e) is the labor earnings
FnsToEvaluate.assets=@(h,aprime,a,z,e) a; % a is the current asset holdings
```

- FnsToEvaluate, create names and equations.
- Note: first inputs are action space, same as ReturnFn. Everything after is understood as parameters.

Basic Life-Cycle Model 4

- Done!
- Code: WorkshopModel4.m (and WorkshopModel4_ReturnFn.m)

Exercise + Break

- You should modify WorkshopModel4.m (and WorkshopModel4_ReturnFn.m) to solve model with i.i.d (e) but without markov (z) shocks.

$$
\begin{aligned}
V(a,e,j) \quad = \quad & \max_{h,c,aprime} \frac{c^{1-\sigma}}{1-\sigma} - \psi \frac{h^{1+\eta}}{1+\eta} \\
& \qquad + s_j \beta E[V(aprime, eprime, j+1)] \\
& \text{if } j < Jr: \; c + aprime = (1+r)a + w\kappa_j h exp(e) \\
& \text{if } j >= Jr: \; c + aprime = (1+r)a + pension \\
& 0 \leq h \leq 1, aprime \geq 0 \\
& e \sim N(0, \sigma_e)
\end{aligned}
$$

Solution is given as WorkshopExercise1.m (and WorkshopExercise1_ReturnFn.m). But try solve without looking at solution.

Life-Cycle Models: Exogenous shocks

- Can have up to four of each of z and e.
  And *semiz* which are explained next slide.
  Action space with, e.g., two z and two e becomes $(d, aprime, a, z1, z2, e1, e2)$.
- Can have grids and probabilites depend on age.
  Can also be created by parameterized functions.
- Can use joint-grids, and so can have correlated shocks.
- VFI Toolkit contains many discretization methods: AR(1), AR(1) with gaussian mixture, VAR(1), and versions for 'non-stationary life-cycle' processes.

See Intro to Life-Cycle Models: Appendix A.

Life-Cycle Models: Exogenous shocks

- Semi-exogenous states.
- *semiz* probabilities depend on a decision variable.
- Create $vfoptions.n\_semiz$, $vfoptions.semiz\_grid$.
- Create $vfoptions.SemiExoStateFn$ for the transition probabilities, will be a function of $(semiz, semizprime, d, \ldots)$.
  Put them in *simoptions* as well.
- Uses: endogenous fertility, search labor, years owned house, endogenous education level.
- Action space: $(d, aprime, a, semiz, z, e, \ldots)$ [*semiz* always comes after a, before z]
- Explain *semiz* vars: semiz vars concept

See Intro to Life-Cycle Models: Life-Cycle Model 29, and Appendix A.

Life-Cycle Models: Permanent Types

- Permanent Types: agents with permanent differences.
- Examples
  - Different values of a parameter.
  - Different exogenous shock processes.
  - Different utility functions.
  - Even that one is finite-horizon and one infinite-horizon.
  - Essentially, anything in toolkit can differ between the permanent types.
- Two ways to set up:
  - Number of permanent types: $N\_i = 2$, $Params.beta = [0.8, 0.95]$
  - Names of permanent types: $Names\_i = \{'patient','impatient'\}$
    $Params.beta.impatient = 0.8$; $Params.beta.patient = 0.95$
- VFI Toolkit commands are same in both cases. And all output uses the names. E.g., $V.patient$ and $V.impatient$.
  If N_i, names are allocated as $ptype001$, $ptype002$, etc.
- All model stats reported as aggregate and conditional on ptype.
  E.g, $AllStats.earnings.Mean$, $AllStats.earnings.patient.Mean$ and $AllStats.earnings.impatient.Mean$.

See Intro to Life-Cycle Models: Life-Cycle Models 24, 25, 27.

Life-Cycle Models: Permanent Types, Number

- Permanent Types: Example 1: earnings fixed effect, only difference is α_i.

$$
\begin{aligned}
V_i(a,z,j) &= \max_{h,c,aprime} \frac{c^{1-\sigma}}{1-\sigma} - \psi \frac{h^{1+\eta}}{1+\eta} \\
&\quad + s_j \beta E[V_i(aprime, zprime, j+1)|z] \\
&\text{if } j < Jr: \; c + aprime = (1+r)a + w\kappa_j h\, exp(z + \alpha_i) \\
&\text{if } j >= Jr: \; c + aprime = (1+r)a + pension \\
&0 \leq h \leq 1, aprime \geq 0 \\
&z' = \rho_z z + \epsilon', \quad \epsilon \sim N(0, \sigma_{z,\epsilon}) \\
&\alpha_i \sim N(0, \sigma_\alpha)
\end{aligned}
$$

- Essentially three changes to code:
- 1. Number of permanent types. 2. Set up agent masses. 3. Commands now include '*_PType*'.
- Code: WorkshopModel5.m (uses WorkshopModel5_ReturnFn.m)

All outputs are now 'by ptype' (look at V, Policy, and also at AllStats).

- Permanent Types: Example 1: earnings fixed effect, α_i.
- 1. Number of permanent types.

```
%% Model action and state-space
n_d=21;
n_a=201; % number of grid points for our
    endogenous state
n_z=9;
N_j=81; % periods, represent ages 20 to 100

N_i=5; % number of permanent types
```

Life-Cycle Models: Permanent Types, Number

- Permanent Types: Example 1: earnings fixed effect, α_i.
- 2. Set up agent masses.

```
% Assume the fixed-effect alpha_i is normally distributed
% Discretize fixed-effect using Farmer-Toda method
[alpha_grid,pi_alpha]=discretizeAR1_FarmerToda(0,0,Params.sigma_alpha,N_i);

% The grid is the values for alpha
Params.alpha_i=alpha_grid;

% The dist we put in the parameter structure
Params.alpha_dist=pi_alpha;
% Name of the parameter which is masses of permanent types
PTypeDistParamNames={'alpha_dist'};
```

Life-Cycle Models: Permanent Types, Number

- Permanent Types: Example 1: earnings fixed effect, α_i.
- 3. Commands now include '_*PType*'.

```
[V, Policy]=ValueFnIter_Case1_FHorz_PType(n_d,
    n_a,n_z,N_j, N_i, d_grid, a_grid, z_grid,
    pi_z, ReturnFn, Params,
    DiscountFactorParamNames, vfoptions);
% Note: _PType, and change in inputs

StationaryDist=StationaryDist_Case1_FHorz_PType(
    jequaloneDist,AgeWeightsParamNames,
    PTypeDistParamNames,Policy,n_d,n_a,n_z,N_j,
    N_i,pi_z,Params,simoptions);
% Note: _PType, and change in inputs

% etc.
```

- Code: WorkshopModel5.m (uses WorkshopModel5_ReturnFn.m)

We also made small change to ReturnFn (to include α_i).

Life-Cycle Models: Permanent Types, Number

- Permanent Types: Example 1: earnings fixed effect, α_i.
- Code: WorkshopModel5.m (uses WorkshopModel5_ReturnFn.m)
- Run code: look at V, StationaryDist, and AgeConditionalStats.
  Note: StationaryDist contains the conditional-on-permanent-type dists (so mass 1) together with the masses of each permanent type (StationaryDist.ptweights).
  Note: AgeConditionalStats.earnings.Mean, also AgeConditionalStats.earnings.ptype001.Mean. Automatically computes both conditional-on-ptype, and the unconditional-on-ptype for stats.

- Permanent Types: Example 2: two households, discount factor, β_i.

$$
\begin{aligned}
V_{i}(a,z,j) \quad &= \quad \max_{h,c,aprime} \frac{c^{1-\sigma}}{1-\sigma} - \psi\frac{h^{1+\eta}}{1+\eta} \\
&\qquad + s_j\beta_i E[V_{i}(aprime, zprime, j+1)|z] \\
&\text{if } j < Jr: \; c + aprime = (1+r)a + w\kappa_j h\, exp(z) \\
&\text{if } j >= Jr: \; c + aprime = (1+r)a + pension \\
&0 \leq h \leq 1, aprime \geq 0 \\
&z' = \rho_z z + \epsilon', \quad \epsilon \sim N(0,\sigma_{z,\epsilon})
\end{aligned}
$$

- Essentially three changes to code:
- 1. Names of permanent types. 2. Set up agent masses. 3. Commands now include '_PType'.
- Code: WorkshopModel6.m (still uses WorkshopModel3_ReturnFn.m)

All outputs are now 'by ptype' (look at V, Policy, and also at AllStats).

Life-Cycle Models: Permanent Types, Number

- Permanent Types: Example 2: discount factor, β_i.
- 1. Name of permanent types.

```
%% Model action and state-space
n_d=21;
n_a=201; % number of grid points for our
    endogenous state
n_z=9;
N_j=81; % periods, represent ages 20 to 100

Names_i={'patient','impatient'}; % names of
    permanent types
```

Life-Cycle Models: Permanent Types, Number

- Permanent Types: Example 2: discount factor, β_i.
- 2. Set up agent masses.

```
%% Permanent types: values and masses
Params.beta.patient=0.99;
Params.beta.impatient=0.9;
% Use the names of the permanent types to set
    different values for them

% Tell VFI Toolkit the name of the parameter
    which is the masses of the permanent types
PTypeDistParamNames={'massofptypes'};
% Set the masses
Params.massofptypes=[0.5,0.5]; % Note: ordering
    follows that in Names_i
```

Life-Cycle Models: Permanent Types, Number

- Permanent Types: Example 2: discount factor, β_i.
- 3. Commands now include '*_PType*'.

```
[V, Policy]=ValueFnIter_Case1_FHorz_PType(n_d,
    n_a,n_z,N_j, N_i, d_grid, a_grid, z_grid,
    pi_z, ReturnFn, Params,
    DiscountFactorParamNames, vfoptions);
% Note: _PType, and change in inputs

StationaryDist=StationaryDist_Case1_FHorz_PType(
    jequaloneDist,AgeWeightsParamNames,
    PTypeDistParamNames,Policy,n_d,n_a,n_z,N_j,
    N_i,pi_z,Params,simoptions);
% Note: _PType, and change in inputs

% etc.
```

- Code: WorkshopModel6.m (still uses WorkshopModel3_ReturnFn.m)

Note: Step 3 is same with 'Numbers' as it was with 'Names'

Note: α_i had to go in ReturnFn, but β_i does not. Hence the ReturnFn here is just same as it was for Workshop Model 3.

Life-Cycle Models: Permanent Types, Number

- Permanent Types: Example 2: discount factor, β_i.
- Code: WorkshopModel6.m (still uses WorkshopModel3_ReturnFn.m)
- Run code: look at V, StationaryDist, and AgeConditionalStats.
  Note: Uses the *Names_i* we set: e.g., V.patient, V.impatient.
  Note: StationaryDist contains the conditional-on-permanent-type dists (so mass 1) together with the masses of each permanent type (StationaryDist.ptweights).
  Note: AgeConditionalStats.earnings.Mean, also AgeConditionalStats.earnings.patient.Mean. Automatically computes both conditional-on-ptype, and the unconditional-on-ptype for stats.

Life-Cycle Models: Permanent Types

- We have barely scratched the surface of what Permanent Types can do.
- Can have different parameters, shocks, grids, ReturnFn, etc.
  E.g., ReturnFn.ptype001 and ReturnFn.ptype002.
- Can even have, e.g., finite-horizon households together with infinite-horizon firms.
- In principle, you can set anything to differ by permanent type.

Life-Cycle Models: Analyse Model

- We saw AllStats and AgeConditionalStats.
- What other model analysis can we do?
- Let's reuse Workshop Model 3 to see different analysis we can do.
- Code: WorkshopModel7.m (still uses WorkshopModel3_ReturnFn.m)
- First half is identical to WorkshopModel3.m, skip to line 117ish.
- Note: All of this also works for permanent types.

Life-Cycle Models: Analyse Model

- Can simulate panel data: same setup (FnsToEvaluate), different command $SimPanelValues = SimPanelValues\_FHorz\_Case1()$.
- Conditional stats: $simoptions.conditionalrestrictions$ can setup a function in same way we do FnsToEvaluate, then when you call AllStats or LifeCycleProfiles, it will also report moments conditional on this function (function must be binary, e.g. $a > 0$)
  See: explain conditionalrestrictions.
- Faster stats: $simoptions.whichstats = ones(7, 1)$ is default. By setting some zeros, you can e.g., skip lorenz curve and quantiles which take most of the time.
  See: explain whichstats.
  Really this is for when you want write a custom code that will need to use AllStats or LifeCycleProfiles a large number of times. If you are using them once you probably don't care about the runtime to calculate a few unnecessary stats.

All of this applies equally to permanent types.

- Other tools for analysing model:
- PolicyInd2Vals: get values for optimal policy (*Policy* just contains indexes).
- ValuesOnGrid: get values for FnsToEvaluate (internally, toolkit combines these with StationaryDist for things like AllStats).
- Age bins: *simoptions.agegroupings*, by default LifeCycleProfiles is giving 1-period age bins. You can set, e.g., 5yr age bins, or working age and retirement as two bins.
  Nice if you want, e.g., gini coeff of earnings for the working age population (in a model that includes retirement periods).
- Tip: If you are doing lots of model analysis, *simoptions* can get messy. So often I will either do *rmfield*() to keep *simoptions* clean, or I will have *simoptions* and *simoptions*2.

Life-Cycle Models: Analyse Model

- Let's reuse Workshop Model 3 to see different analysis we can do.
- Code: WorkshopModel7.m (still uses WorkshopModel3_ReturnFn.m)
- First half is identical to WorkshopModel3.m, skip to line 117ish.
- Note: All of this also works for permanent types.

- Everything we have seen so far is what I think of as the 'core' features.
- They work in combination with everything else we will see.
- So all shock types, permanent types, and model analysis, will work with all the more advanced features.

References I

Robert Kirkby. VFI toolkit, v2. *Zenodo*, 2022. doi: https://doi.org/10.5281/zenodo.8136790.

- Can write Bellman equation like
$$V(a,z,e)=\max_{d,a'} F(d,a',a,z,e)+s_j\beta E[V(a',z',e')|z]$$
- ReturnFn will be a function that plays role of $F(d,a',a,z,e)$
- e is an i.i.d exogenous state (so will have probability distriubtion pi_e).

Back

- Can write Bellman equation like

$$V(a, semiz, z) = \max_{d,a'} F(d, a', a, semiz, z) + s_j \beta E[V(a', semiz', z')|semiz, z, d]$$

- ReturnFn will be a function that plays role of $F(d, a', a, semiz, z)$
- *semi* is an semi-exogenous markov state (so will have probability transitions that depend on $(semiz, semizprime, d..)$ and a set using $vfoptions.SemiExoStateFn$.
- Roughly, *semiz* is a markov state, but where depending on value of the decision variable, you get different transition matrix.
- I here simplify and have d in everything, in practice often you can split d in $d1$ and $d2$, with both in ReturnFn, but only $d2$ determine semi-exo state.

Back