This document describes the models in the folder `/ConsumptionSaving` as of 6/25/16.

1 ConsIndShockModel.py

Defines consumption-saving models whose agents have CRRA utility over a unitary consumption good, geometric discounting, and who face idiosyncratic shocks to income.

1.1 Perfect Foresight

Consider an agent with CRRA utility over consumption, who discounts future utility at a constant rate per period and has no bequest motive. His problem can be written as:

$$\begin{aligned}
V_t(M_t) &= \max_{C_t} \mathrm{u}(C_t) + \beta \cancel{D}_{t+1} \mathbb{E}[V_{t+1}(M_{t+1})], \\
A_t &= M_t - C_t, \\
M_{t+1} &= \mathsf{R} A_t + Y_{t+1}, \\
Y_{t+1} &= \Gamma_{t+1} Y_t, \\
\mathrm{u}(C) &= \frac{C^{1-\rho}}{1-\rho}.
\end{aligned}$$

The model can be normalized by current income (which is also permanent income in this model) by defining lower case variables as their upper case version divided by Y_t:

$$\begin{aligned}
\mathrm{v}_t(m_t) &= \max_{c_t} \mathrm{u}(c_t) + \beta \cancel{D}_{t+1} \mathbb{E}[\mathrm{v}_{t+1}(m_{t+1})], \\
a_t &= m_t - c_t, \\
m_{t+1} &= (\mathsf{R}/\Gamma_{t+1}) a_t + 1, \\
\mathrm{u}(c) &= \frac{c^{1-\rho}}{1-\rho}.
\end{aligned}$$

An individual agent's model is thus characterized by values of ρ, β, and R along with sequences $\{\Gamma_t\}_{t=1}^{T}$ and $\{\cancel{D}_t\}_{t=1}^{T}$, with $T = \infty$ possible.

The one period problem for this model is solved by the function `solveConsPerfForesight`, which creates an instance of the class `ConsPerfForesightSolver`. The class `PerfForesightConsumerType` extends `AgentType` to represent agents in this model. The concordance between model variables and their code equivalents is as follows.

Var	Description	Code
ρ	Coefficient of relative risk aversion	`CRRA`
β	Intertemporal discount factor	`DiscFac`
R	Risk free interest factor	`Rfree`
$\cancel{D}$	Survival probability	`LivPrb`
Γ	Permanent income growth factor	`PermGroFac`

These are the only five parameters that an instance of `PerfForesightConsumerType` must have in order to use its `solve` method. Note that `LivPrb` and `PermGroFac` are assumed to be time-varying, so they should be input as a list. Each element of the `solution` attribute will be an instance of `ConsumerSolution` with the following attributes:

Var	Description	Code
$\mathrm{c}(\cdot)$	Noramlized consumption function	`cFunc`
$\mathrm{v}(\cdot)$	Normalized value function	`vFunc`
$\mathrm{v}'(\cdot)$	Normalized marginal value function	`vPfunc`
$\underline{m}$	Mininum normalized market resources	`mNrmMin`
h	Normalized human wealth	`hNrm`
$\overline{\kappa}$	Maximum marginal propensity to consume	`MPCmax`
$\underline{\kappa}$	Minimum marginal propensity to consume	`MPCmin`

In the perfect foresight model, the consumption function is linear, so the maximum and minimum MPC are equal. Each of the functions takes normalized market resources m as an argument, and they only defined on the domain $m \geq \underline{m} = -h$.

1.2 Permanent and Transitory Idiosyncratic Shocks

Consider an agent with CRRA utility over consumption, who discounts future utility at a constant rate per period and has no bequest motive. He foresees that he will experience shocks to his income that are fully transitory or fully permanent. Using the normalization above, his problem can be written as:

$$
\begin{aligned}
\mathrm{v}_t(m_t) &= \max_{c_t} \mathrm{u}(c_t) + \beta \not{D}_{t+1} \mathbb{E}[\mathrm{v}_{t+1}(m_{t+1})], \\
a_t &= m_t - c_t, \\
a_t &\geq \underline{a}, \\
m_{t+1} &= \mathsf{R}/(\Gamma_{t+1}\psi_{t+1}) a_t + \theta_{t+1}, \\
\theta_t \sim F_{\theta t}, & \qquad \psi_t \sim F_{\psi t}, \quad \mathbb{E}[F_{\psi t}] = 1 \\
\mathrm{u}(c) &= \frac{c^{1-\rho}}{1-\rho}.
\end{aligned}
$$

That is, this agent is identical to the perfect foresight agent except that his income is subject to permanent (ψ) and transitory (θ) shocks to income, and he might have an artificial borrowing constraint $\underline{a}$.

The one period problem for this model is solved by the function `solveConsIndShock`, which creates an instance of the class `ConsIndShockSolver`. The class `IndShockConsumerType` extends `PerfForesightConsumerType` to represent agents in this model. To construct an instance of this class, several additional parameters must be passed to the constructor. Note that most of these parameters are *indirect* inputs to the consumer's model: they are used to construct direct inputs to the one period problem. The concordance between the model and code is as follows:

Var	Description	Code
(none)	Minimum of "assets above minimum" grid	`aXtraMin`
(none)	Maximum of "assets above minimum" grid	`aXtraMax`
(none)	Number of points in "assets above minimum" grid	`aXtraCount`
(none)	Additional values for the "assets above minimum" grid	`aXtraExtra`
(none)	Degree of exponential nesting for assets grid	`exp_nest`
N_θ	Number of discrete values in transitory shock distribution	`TranShkCount`
N_ψ	Number of discrete values in permanent shock distribution	`PermShkCount`
σ_θ	Standard deviation of log transitory shocks	`TranShkStd`
σ_ψ	Standard deviation of log permanent shocks	`PermShkStd`
$\mho$	Unemployment probability in working period	`UnempPrb`
$\mho_{ret}$	"Unemployment" probability in retirement period	`UnempPrbRet`
$\underline{\theta}$	Transitory income when unemployed in working period	`IncUnemp`
$\underline{\theta}_{ret}$	Transitory income when "unemployed" in retired period	`IncUnempRet`
τ	Marginal income tax rate	`tax_rate`
T_{ret}	Period of retirement; number of working periods	`T_retire`
$\underline{a}$	Artificial borrowing constraint	`BoroCnstArt`
(none)	Indicator for whether `cFunc` should use cubic splines	`CubicBool`
(none)	Indicator for whether `vFunc` should be computed	`vFuncBool`
T	Total number of (non-terminal) periods in sequence	`T_total`
(none)	Number of agents of this type	`Nagents`

The first five attributes in the table above are used to construct the "assets above minimum" grid `aXtraGrid`, an input for `solveConsIndShock`.[1] The next ten attributes specify an assumed form for the income distribution $(F_{\psi t}, F_{\theta t})$. Both permanent and transitory shocks are lognormally distributed, and with a point mass in the transitory distribution representing unemployment. Further, the sequence of periods is broken into two parts, "working" and "retired" to allow for a different income process in retirement.[2] The attributes `PermShkStd` and `TranShkStd` are thus lists of the (log) standard deviation of shocks period-by-period.

Like the assets grid, the specification of the income process can be changed with little difficulty. No matter what form is used, the relevant direct input to `solveConsIndShock` is `IncomeDstn`, a finite discrete approximation to the true income process. This attribute

[1]In the current configuration, the grid is multi-exponentially spaced given minimum, maximum, number of gridpoints, and degree of exponential nesting (with additional values to force into the grid with `aXtraExtra`). It is simple to replace this grid with another by changing the function `makeAssetsGrid`.

[2]Permanent and transitory shocks are turned off during retirement, other than the possibility of "unemployment", representing (say) a temporary failure of the retirement benefit system.

is specified as a list with three elements: an array of probabilities (that sum to 1), an array of permanent income shocks, and an array of transitory income shocks.

The artificial borrowing constraint imposes a restriction on assets at the end of the period; it can be set to `None` to turn off the constraint (i.e. only the "natural" borrowing constraint will be used). The attributes `CubicBool` and `vFuncBool` should be set to `True` or `False`, as their name implies. The solver can construct a linear or cubic spline interpolation of the consumption function; cubic interpolation is slower but more accurate at any number of gridpoints. The value function is not strictly necessary to compute during solution and carries a computational burden, so it can be turned off with `vFuncBool=False`. The number of agents of this type `Nagents` is irrelevant during solution and is only used during simulation (when *ex-post* heterogeneity emerges within the *ex-ante* homogeneous type).

The `solve` method of `IndShockConsumerType` will populate the `solution` attribute with a list containing instances of `ConsumerSolution`. Each of these instances has all the elements listed above in the perfect foresight section plus the attribute `vPPfunc` (representing $\mathrm{v}''(m)$) if `CubicBool=True`.[3]

1.3 Different Interest Rate on Borrowing vs Saving

Consider an agent identical to the "idiosyncratic shocks" model above, except that his interest factor differs depending on whether he borrows or saves on net. His problem is the same as the one above, with a simple addition:

$$\mathsf{R} = \begin{cases} \mathsf{R}_{boro} & \text{if } a_t < 0 \\ \mathsf{R}_{save} & \text{if } a_t > 0 \end{cases}, \qquad \mathsf{R}_{boro} \geq \mathsf{R}_{save}.$$

The one period problem for this model is solved by `solveConsKinkedR`, which creates an instance of `ConsKinkedRsolver`. The class `KinkedRconsumerType` extends `IndShockConsumerType` to represent agents in this model. The attributes required to specify an instance of `KinkedRconsumerType` are the same as `IndShockConsumerType` except that `Rfree` *should not* be included, instead replaced by values of `Rboro` and `Rsave`. The "kinked R" solver is not yet compatible with cubic spline interpolation for `cFunc`; if the `solve` method is run with `CubicBool=True`, it will throw an exception.[4]

The `solve` method of `KinkedRconsumerType` populates the `solution` attribute with a list of `ConsumerSolution` instances, in the same format as the idiosyncratic shocks model.

[3] `vFunc` will be a placeholder function of the class `NullFunc` if `vFuncBool=False`.

[4] This is an item that is ripe for development by an outside contributor.

2 ConsPrefShockModel.py

Defines consumption-saving models whose agents have CRRA utility over a unitary consumption good, geometric discounting, who face idiosyncratic shocks to income and to their utility or preferences.

2.1 Multiplicative Shocks to Utility

Consider an agent with a very similar problem to that of the "idiosyncratic shocks" model in the preceding section, except that he receives an iid multiplicative shock to his utility at the beginning of each period, before making the consumption decision. This model can be written in Bellman form as:

$$
\begin{aligned}
\mathrm{v}_t(m_t, \eta_t) &= \max_{c_t} \eta \cdot \mathrm{u}(c_t) + \beta \cancel{D}_{t+1} \mathbb{E}[\mathrm{v}_{t+1}(m_{t+1}, \eta_{t+1})] \\
a_t &= m_t - c_t \\
a_t &\geq \underline{a} \\
m_{t+1} &= \mathsf{R}/(\Gamma_{t+1}\psi_{t+1})a_t + \theta_{t+1} \\
\theta_t \sim F_{\theta t}, & \qquad \psi_t \sim F_{\psi t}, \quad \mathbb{E}[F_{\psi t}] = 1 \\
\mathrm{u}(c) &= \frac{c^{1-\rho}}{1-\rho}, \qquad \eta_t \sim F_{\eta t}.
\end{aligned}
$$

The one period problem for this model is solved by the function `solveConsPrefShock`, which creates an instance of `ConsPrefShockSolver`. The class `PrefShockConsumerType` is used to represent agents in this model. The attributes required to construct an instance of this class are the same as for `IndShockConsumerType` above, but with three additions:

Var	Description	Code
N_η	Number of discrete points in "body" of preference shock distribution	`PrefShkCount`
N_η^{tail}	Number of discrete points in "tails" of preference shock distribution	`PrefShk_tail_N`
σ_η	Log standard deviation of multiplicative utility shocks	`PrefShkStd`

These attributes are indirect inputs to the problem, used during instantiation to construct the `PrefShkDstn`, an input to `solveConsPrefShock`. The tails of the preference shock distribution matter a great deal for the accuracy of the solution and are underrepresented by the default equiprobable discrete approximation (unless a very large number of points are used). To fix this issue, the attribute `PrefShk_tail_N` specifies the number of points in each "augmented tail" section of the preference shock discrete approximation.[5] The standard deviation of preference shocks might vary by period, so `PrefShkStd` should

[5]See documentation for `HARKutilities.approxLognormal` for more details.

be input as a list. The "preference shock" solver is not yet compatible with cubic spline interpolation for the consumption function and will throw an exception if `CubicBool=True`.

The `solve` method of `PrefShockConsumerType` populates the `solution` attribute with a list of `ConsumerSolution` instaces. These single-period-solution objects have the same attributes as the "idiosyncratic shocks" models above, but the attribute `cFunc` is defined over the space of (m_t, η_t) rather than just m_t. The value function `vFunc` and marginal value `vPfunc`, however, are defined *only* over m_t, as they represent expected (marginal) value *just before* the preference shock η_t is realized:[6]

$$\begin{aligned}\overline{\mathrm{v}}_t(m_t) &= \int_0^\infty \mathrm{v}(m_t, \eta) dF_{\eta t}(\eta), \\ \overline{\mathrm{v}}_t'(m_t) &= \int_0^\infty \mathrm{v}'(m_t, \eta) dF_{\eta t}(\eta).\end{aligned}$$

2.2 Utility Shocks and Different Interest Rates

Consider an agent with idiosyncratic shocks to permanent and transitory income and multiplicative shocks to utility *and* faces a different interest rate on borrowing vs saving. This agent's model is identical to that of the "preference shock" consumer in section 2.1, with the addition of the interest rate rule from the "kinked R" consumer in section 1.3.

The one period problem of this combination model is solved by the function `solveConsKinkyPref`, which creates an instance of `ConsKinkyPrefSolver`. The class `KinkyPrefConsumerType` represents agents in this model. As you will see in `ConsPrefShockModel.py`, there is *very* little new code required to program this model: the solver and consumer classes each inherit from both `KinkedR` and `PrefShock` and only need a trivial constructor function to rectify the differences between the two. This is a good demonstration of the benefit of HARK's object-oriented approach to solution methods: it is sometimes trivial to combine two models to make a new one.

The attributes required to properly construct an instance of `KinkyPrefConsumerType` are the same as for `PrefShockConsumerType` except that (like the "kinked R" parent model) `Rfree` should not be replaced with `Rboro` and `Rsave`. Like both of its parents, `KinkyPref` is not yet compatible with cubic spline interpolation of the consumption function.

[6] Particularly in the case of `vPfunc`, this is the object of interest for solving the preceding period.

3 ConsMarkovModel.py

blah blah

4 ConsAggShockModel.py

blah blah

5 TractableBufferStockModel.py

Defines the "tractable buffer stock" model from Chris Carroll's lecture notes.

5.1 Tractable Buffer Stock

Consider a consumer with CRRA utility who faces only a single, very specific risk: that he will become permanently unemployed and receive no income until the end of time. Otherwise, he faces an infinite horizon problem with a steady stream of income that grows by a fixed factor each period, and earns a constant rate of return on assets retained betweed periods. His model when still employed can be written in Bellman form as:[7]

$$\begin{aligned}
\mathrm{v}^e(m_t) &= \max_{c_t} \mathrm{u}(c_t) + \beta\left((1-\mho)\mathrm{v}^e(m^e_{t+1}) + \mho \mathrm{v}^u(m^u_{t+1})\right) \\
a_t &= m_t - c_t \\
m^e_{t+1} &= (\mathsf{R}/\widehat{\Gamma})a_t + 1, \qquad \widehat{\Gamma} = \Gamma/(1-\mho) \\
m^u_{t+1} &= (\mathsf{R}/\widehat{\Gamma})a_t.
\end{aligned}$$

His model while unemployed is simply:

$$\begin{aligned}
\mathrm{v}^u(m_t) &= \max_{c_t} \mathrm{u}(c_t) + \beta \mathrm{v}^u(m^u_{t+1}) \\
a_t &= m_t - c_t \\
m^u_{t+1} &= (\mathsf{R}/\widehat{\Gamma})a_t.
\end{aligned}$$

This model is solved by the class `TractableConsumerType` when its `solve()` method is invoked. An instance of this class is specified by the five parameters in the table below:

Var	Description	Code
ρ	Coefficient of relative risk aversion	`CRRA`
β	Intertemporal discount factor	`DiscFac`
R	Interest factor on assets	`Rfree`
Γ	Permanent income growth factor	`PermGroFac`
$\mho$	Probability of becoming unemployed	`UnempPrb`

[7] For technical / teaching reasons, permanent income growth while employed is "risk compensated" so that human wealth does not vary with the unemployment probability.