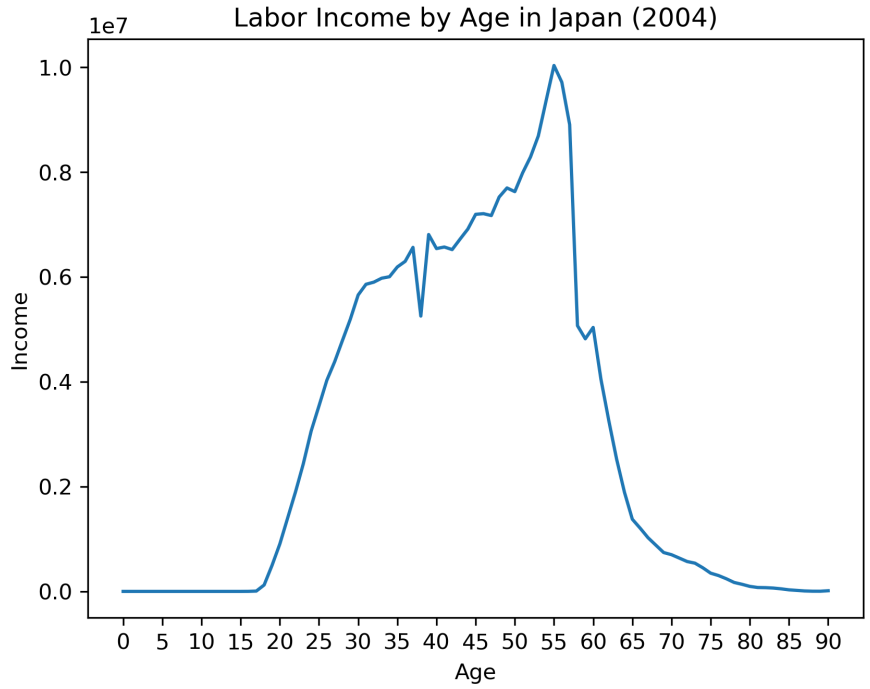
3 Large-scale OLG

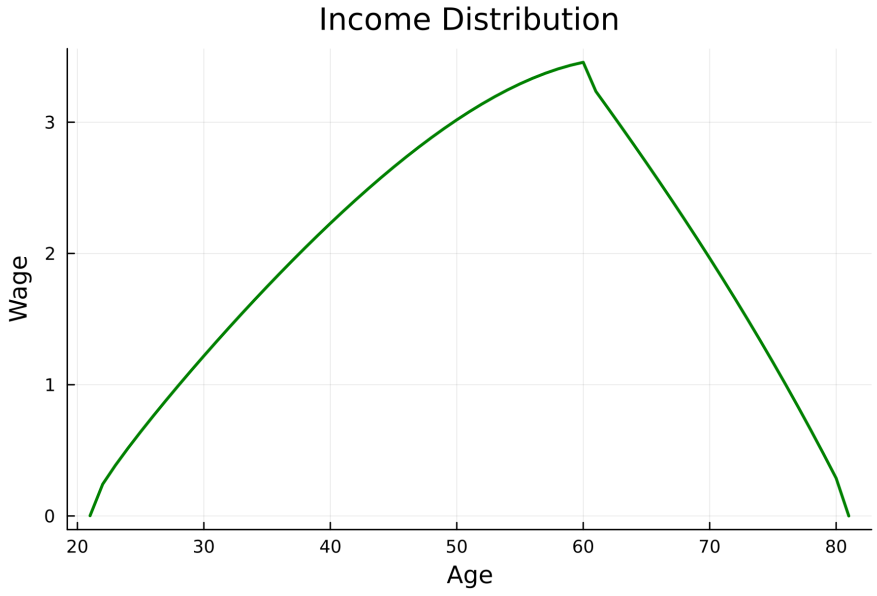
Premise

1. Agents are heterogeneous in age.
2. There is retirement. Agents do not live forever.
3. Population is constant forever.

We want to reproduce a life cycle profile such as this



Our goal is to capture similar qualitative features in the model.



3.1 The Model

Building Blocks

Age structure: The representative household's life span is 60, such that

$$T + TR = 40 + 20 = 60.$$

where T is the working length and TR is the retirement length. Labor supply n_t^s follows

$$l_t^s = 1 - n_t^s \text{ for } t \in \{1, 2, \ldots, 40\}, \tag{28}$$
$$l_t^s = 1 \text{ for } t \in \{41, 42, \ldots, 60\}. \tag{29}$$

where l_t^s is leisure. After T years, retirement is mandatory. The agent's maximization problem is

$$\sum_{s=1}^{T+TR} \beta^{s-1} u(c_{s+t-1}^s, l_{t+s-1}^s). \tag{30}$$

where β is the discount factor. The instantaneous utility function:

$$u(c, l) = \frac{((c + \psi) l^\gamma)^{1-\eta} - 1}{1 - \eta}. \tag{31}$$

An agent is born without wealth and leaves no bequests upon death, thus $k_t^1 = k_t^{61} = 0$. The real budget constraint of the working agent is given by

$$k_{t+1}^{s+1} = (1 + r_t) k_t^s + (1 - \tau_t) w_t n_t^s - c_t^s \text{ for } s = 1, \ldots, T. \tag{32}$$

where r_t, w_t are the interest and wage rates, while τ is the social security contribution tax. Once retired, the agents receive public pensions b and no labor earnings. The budget constraint for a retiree is

$$k_{t+1}^{s+1} = (1 + r_t) k_t^s + b - c_t^s \text{ for } s = T + 1, \ldots, TR. \tag{33}$$

Production is Cobb-Douglas technology:

$$Y_t = N_t^{1-\alpha} K_t^\alpha.$$

Let $\delta \in [0, 1]$ be the depreciation rate. The factor prices are

$$w_t = (1 - \alpha) K_t^\alpha N_t^{1-\alpha},$$
$$r_t = \alpha K_t^{\alpha-1} N_t^{-\alpha} - \delta.$$

Its budget is balanced every period such that

$$\tau w_t N_t = \frac{TR}{T + TR} b.$$

Equilibrium

To derive the equilibrium, we need to solve the household problem.

Exercise 8. Solve a young household's problem by maximizing (30), using the functional form at (31), with respect to (28) and (32) by doing the following steps

1. Write the Lagrangian.
2. Derive the FOC for c and l.
3. Derive the Euler equation.

Solving a retiree household's problem by maximizing (30), using the functional form at (31), with respect to (29) and (33).

1. From (29), we know that $n_t^s = 0$.
2. Derive the Euler equation

Answer 1. The FOC:

$$\frac{u_l(c_t^s, l_t^s)}{u_c(c_t^s, l_t^s)} = \gamma \frac{c_t^s + \psi}{l_t^s} = (1 - \tau_t) w_t. \tag{34}$$

The Euler:

$$\frac{1}{\beta} = \frac{u_c(c_{t+1}^{s+1}, l_{t+1}^{s+1})}{u_c(c_t^s, l_t^s)}(1 + r_{t+1}) = \frac{(c_{t+1}^{s+1} + \psi)^{-\eta}(l_{t+1}^{s+1})^{\gamma(1-\eta)}}{(c_t^s + \psi)^{-\eta}(l_t^s)^{\gamma(1-\eta)}}(1 + r_{t+1}). \tag{35}$$

We can also represent the households' problem recursively. Let $V^s(k_t^s, K_t, N_t)$ be the value of the objective function of s-year old agent with wealth k_t^s, K_t, N_t. It is defined as the solution to the dynamic program

$$V^s(k_t^s, K_t, N_t) = \begin{cases} \max\limits_{k_{t+1}^{s+1}, c_t^s, l_t^s} u(c_t^s, l_t^s) + \beta V^{s+1}(k_{t+1}^{s+1}, K_{t+1}, N_{t+1}) \text{ for } s = 1, \ldots, T. \\ \max\limits_{k_{t+1}^{s+1}, c_t^s} u(c_t^s, 1) + \beta V^{s+1}(k_{t+1}^{s+1}, K_{t+1}, N_{t+1}) \text{ for } s = T+1, \ldots, T+TR-1. \end{cases} \tag{36}$$

subject to Eq.(32) and (33), respectively. Furthermore

$$V^{T+TR}(k_t^{T+TR}, K_t^{T+TR}, N_t^{T+TR}) = u(c^{T+TR}, 1)$$

implying that agents obtain nothing if they save after they die. The recursive formulation is useful for the Value Function Iteration method.

Definition 2. Equilibrium

1. Individual and aggregate behaviors are consistent

$$N_t = \sum_{s=1}^{T+TR} \frac{n_t^s}{T + TR}, \tag{37}$$

$$K_t = \sum_{s=1}^{T+TR} \frac{k_t^s}{T + TR}. \tag{38}$$

   The formulation means that aggregate labor (capital) supply is equal to the sum of supplies of each cohort, weighted by its mass $1/(T + TR) = 1/60$.

2. Goods market clear

$$N_t^{1-\alpha} K_t^{\alpha} = \sum_{s=1}^{T+TR} \frac{c_t^s}{T + TR} + K_{t+1} - (1 - \delta) K_t.$$

3. Given $\{w_t, r_t, b, \tau\}$, the household's problem is solved according to Eq.(34) and (35), the firm's optimization is solved, and the government budget is balanced.

3.2 Steady State

The concept of a steady state can be characterized by a constant distribution of capital stock over generations.

$$\{k_t^s\}_{s=1}^{60} = \{k_{t+1}^s\}_{s=1}^{60} = \{\bar{k}^s\}_{s=1}^{60}.$$

As a consequence, every other variable, such as r, w, b, τ, also becomes a constant.

The computation of the steady states is more complex than Section 1. However, since our functions are well-behaved, we can use the simple iteration method to reach the steady state by following the algorithm below.

1. Make initial guesses of the steady state values of K and N.
2. Compute w, r, τ, b that solve the firm's problem and the government's budget set.
3. Compute the optimal path for consumption, savings, and labor supply by backward iteration.
   (a) We know $k^1 = k^{61} = 0$. Make a guess of k^{60}.
   (b) With k^{61}, k^{60} known, solve for $k^{59}, k^{58}, \ldots, k^2, k^1$.
   (c) Compute k^1:
      i. if $k^1 = 0$. Stop the loop and output the series of $k^1, \ldots, k^{60}$.
      ii. else, if $k^1 \neq 0$, update k^{60} using the secant method.
4. Recompute the new aggregate K and N.
5. If they are the same as the initial guess. Stop. Otherwise, update a new K and N and go back to step 2 until convergence.

Parameters

Parameters	Value
β	0.98
η	2
α	0.36
δ	0.1
γ	2
ψ	0.001

To calculate the tax rate, set the replacement ratio $\xi = 0.3$ such that

$$\xi = \frac{b}{(1-\tau)w\bar{n}}.$$

with $\bar{n}$ is the average labor supply, which equals to $N(T + TR)/T$. Since we want a realistic value, you can set the target values for the steady states of

$$\bar{n} = 0.35, \qquad \tau = \frac{\xi}{2+\xi}, \qquad r = 0.045.$$

Given N and r, we can calculate w and K, then solve for b, which completes step 2.

3.3 Direct Computation

Exercise 9. In this exercise, you are asked to perform direct computation at each age. Note that there are three crucial age thresholds: age 39 (work at 39, work at 40); age 40 (work at 40, retire at 41); age 41 (retire for the rest of life).

1. Write the FOC and Euler equations for the young from age 1 to 39.
2. Write the FOC and Euler for age 40 only.
3. Write the Euler from age 41 to 60.
4. Write 3 functions that solve the decision rules for young age, age 40, and retirees.
5. Next, write 1 iteration, given $k[61], n[61], k[61], n[60]$, perform backward iteration and solve until $n[1], k[1]$.
6. Next, write a program of many iterations as step 5. For each iteration, you calculate the error value and update new guesses of n, k using the second method.
7. Step 6 completes an inner loop. Compress it into a function, taking the guess of N as argument (input), and output 2 arrays: age profile k^s and n^s.
8. Write an outer loop. In each loop, take N as given, run the backward iteration function written in step 6, output the new aggregate N, and then update the new guess if necessary.

Tips

- Because we are solving backward, it is better to use Julia because Julia's index system starts from 1.
- Technically, once the agent retires, he supplies zero labor and only cares about how much to consume and save, which is governed by the Euler equation. Hence, the current capital holding can be solved explicitly.
- For young agents, you need to solve simultaneously a system of 2 equations (FOC and Euler) of 2 variables (consumption and labor). As the model cannot be solved explicitly, use this code snippet to solve:

```
using NLsolve
# solving for x,y -> z, with parameters a,b,c
function system_eq!(F, z, a, b, c)
  x, y = z
  F[1] = x^2 + y^2 - a + 2b
  F[2] = (x * y)^c - a - b^2
end
#specify values of parameters
a = 1.0
b = 2.0
c = 2.0
z_guess = [0.0, 0.0]  # Initial guess for x and y
# Solve the system of equations
result = nlsolve((F, z) -> system_eq!(F, z, a, b, c), z_guess)
# Extract the solution
x_solution = result.zero[1]
y_solution = result.zero[2]
```

Note: The correct guess is very important when applying this solver. The closer it is to the true solutions, the more accuracy it gets. With backward iteration, I recommend taking the previously solved known value as the guess. For example, in Auerbach-Kotlikoff, the guess to solve k^{39} should take the initial guess (for the solver) of k and n from k^{40}, k^{41}, n^{40}.

- You should start with step 3. Once you tested its reliability, write it into a function. After that, write an outer loop for steps 2 to 5.

More on step 3

For backward iteration, since $k^{61} = 0 = k^1$, we only need to calculate the decision rule for $k^{59}, k^{58}, \ldots, k^2$.

For $t = 59, \ldots, 41$: Since the retiree does not work, we only need to deal with the Euler equation to determine k. To calculate the optimal k^{59}, we need the information of k^{60} and k^{61}. Although k^{61} is known, k^{60} is not. Here, we provide the first 2 guesses for the first 2 iterations

$$k_1^{60} = 0.15, k_2^{60} = 0.2.$$

the subscript denotes the i^{th} iteration.

For $t = 40$, you need to find the optimal values of $n^{40}(n^{41}, k^{41})$, $k^{40}(k^{41}, k^{42}, n^{41}, n^{42})$. As this is the last period with a positive labor supply, we have $n^{41} = n^{42} = 0$. Capital holdings k^{41}, k^{42} have been solved in the retiree's problem. Hence, you have 2 equations of 2 unknowns. We can use Julia's 'NLSolve'.

For $t = 39, \ldots, 1$, we will do things similar to the previous steps. Note that when we calculate k^1, if it is different from the tolerance, we must update K and N.

Updating k^{60} using the Secant Method:

$$k_i^{60} = k_{i-1}^{60} - \frac{k_{i-1}^{60} - k_{i-2}^{60}}{k_{i-1}^{1} - k_{i-2}^{1}} k_{i-1}^{1}.$$

More on steps 1 and 5

It is actually sufficient to make only a guess of N since K must scale up with it due to Cobb-Douglas technology. Since there is a target for the steady state N, it is a more appropriate approach than guessing randomly. In the code, *nbar* is the aggregate N.

The update algorithm is simple:

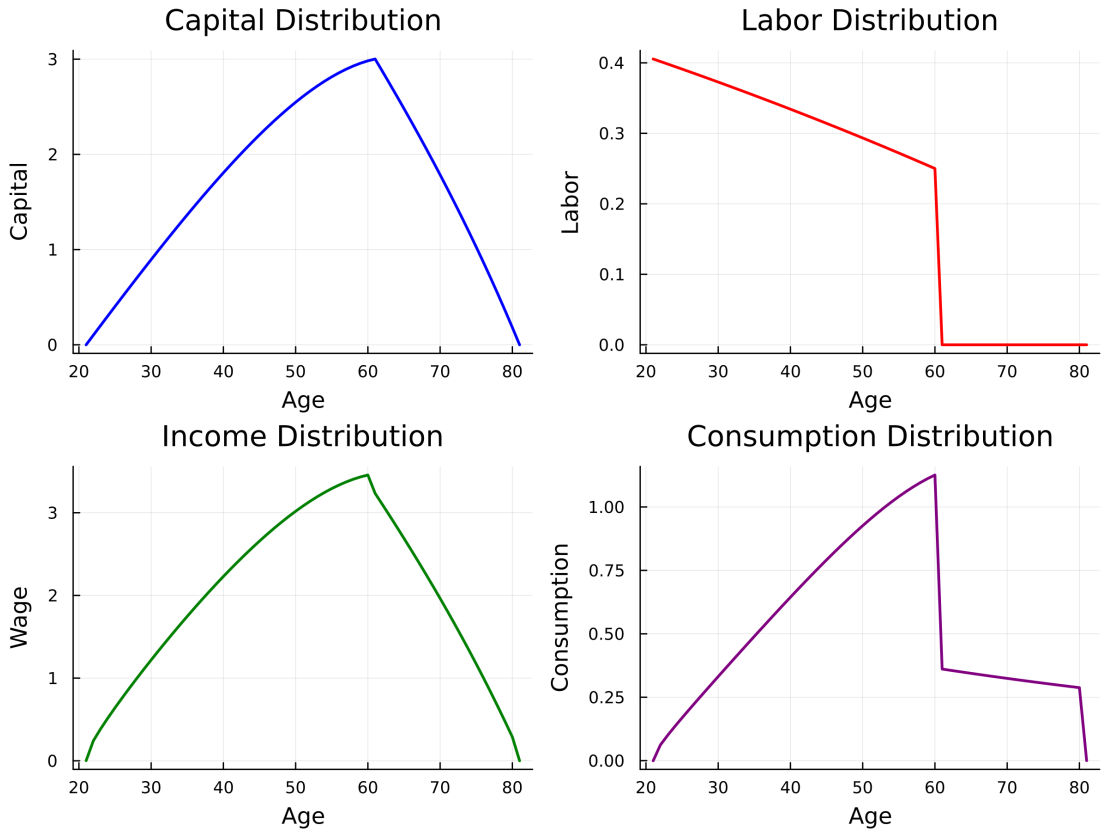
$$N_j^{guess} = \phi N_{j-1}^{guess} + (1 - \phi) N_{j-1}^{out}.$$

with ϕ is the learning rate, j is the j^{th} iteration. N^{out} is the outcome of an iteration taking N^{guess} as an input. The algorithm should update and converge at iteration $\hat{j}$ such that $N_{\hat{j}}^{guess} = N_{\hat{j}}^{out}$.

For the first 2 initial guesses, use $N_1 = 0.15$, $N_2 = 0.25$.

Results

You should find the steady state of K and N as 0.913 and 0.221, respectively.



Further Applications

Two other solution methods exist Value function iteration and projection methods. As they require more advanced programming, this camp does not have enough time to cover them. As indicated in Heer and Maußner (2009), the direct computation is fast, reliable, and accurate. Whenever possible, this method should be applied. You can extend this model with more sophisticated features, such as

- age-specific productivity (Huggett, 1996)
- age-specific survival probabilities (İmrohoroglu et al., 1995; Huggett and Ventura, 1999)
- money holding (Heer et al., 2007)
- government debt (Braun and Joines, 2015)

One limitation of this method is that it is not feasible to solve the model with idiosyncratic uncertainty, such as stochastic autocorrelated productivity shocks. In such an environment, you need to use value function iteration or projection methods.