Expectated vs Realized Income Growth in A Standard Life Cycle Model

This notebook uses the income process in Cocco, Gomes & Maenhout (2005) to demonstrate that estimates of a regression of expected income changes on realized income changes are sensitive to the size of transitory shocks.

We first load some tools from the HARK toolkit.

In [1]:

```
from HARK.ConsumptionSaving.ConsIndShockModel import (
    IndShockConsumerType,
    init_lifecycle,
)

from HARK.Calibration.Income.IncomeTools import (
    parse_income_spec,
    parse_time_params,
    CGM_income,
)

from HARK.datasets.life_tables.us_ssa.SSATools import parse_ssa_life_table
from HARK.datasets.SCF.WealthIncomeDist.SCFDistTools import income_wealth_dists_from_scf
import matplotlib.pyplot as plt
import pandas as pd
from copy import copy
```

We now create a population of agents with the income process of Cocco, Gomes & Maenhout (2005), which is implemented as a default calibration in the toolkit.

In [2]:

```
birth_age = 21
death_age = 66
adjust_infl_to = 1992
income_calib = CGM_income
education = "HS"

# Income specification
income_params = parse_income_spec(
    age_min=birth_age,
    age_max=death_age,
    adjust_infl_to=adjust_infl_to,
    **income_calib[education],
    SabelhausSong=True
)

# We need survival probabilities only up to death_age-1, because survival
# probability at death_age is 1.
liv_prb = parse_ssa_life_table(
    female=True, cross_sec=True, year=2004, min_age=birth_age, max_age=death_age - 1
)

# Parameters related to the number of periods implied by the calibration
time_params = parse_time_params(age_birth=birth_age, age_death=death_age)

# Update all the new parameters
params = copy(init_lifecycle)
params.update(time_params)
# params.update(dist_params)
params.update(income_params)
params.update(
    {
        "LivPrb": liv_prb,
        "pLvlInitStd": 0.0,
        "PermGroFacAgg": 1.0,
        "UnempPrb": 0.0,
        "UnempPrbRet": 0.0,
        "track_vars": ["pLvl", "t_age", "PermShk", "TranShk"],
        "AgentCount": 200,
        "T_sim": 500,
    }
)

Agent = IndShockConsumerType(**params)
Agent.solve()
```

We simulate a population of agents

In [3]:

```
%%capture
# Run the simulations
Agent.initialize_sim()
Agent.simulate()
```

We assume a standard income process with transitory and permanent shocks: The consumer's **p**ermanent noncapital income P grows by a predictable factor Γ and is subject to an unpredictable multiplicative shock $\mathbb{E}_t[\psi_{t+1}] = 1$,

$$P_{t+1} = P_t\Gamma_{t+1}\psi_{t+1},$$

and, if the consumer is employed, actual income Y is permanent income multiplied by a transitory shock $\mathbb{E}_t[\theta_{t+1}] = 1$,

$$Y_{t+1} = P_{t+1}\Gamma_{t+1}\theta_{t+1},$$

Define $y = \log Y, p = \log P$ and similarly for other variables.

Γ_t captures the predictable life cycle profile of income growth (faster when young, slower when old). See our replication of CGM-2005 for a detailed account of how these objects map to CGM's notation.

Now we construct all the necessary inputs to the regressors. The main input is the expected income growth of every agent at every time period, which is given by

$$\mathbb{E}_t[Y_{t+1}/Y_t] = \mathbb{E}_t[\left(\frac{\theta_{t+1}P_t\Gamma_{t+1}\psi_{t+1}}{\theta_t P_t}\right)]$$

$$= \left(\frac{\Gamma_{t+1}}{\theta_t}\right)$$

$$\mathbb{E}_t[y_{t+1} - y_t] = \log\Gamma_{t+1} - \log\theta_t$$

In [4]:

```
from HARK.distribution import calc_expectation

exp = [
    calc_expectation(Agent.IncShkDstn[i], func=lambda x: x[0] * x[1])
    for i in range(Agent.T_cycle)
]
exp_df = pd.DataFrame(
    {
        "exp_prod": exp,
        "PermGroFac": Agent.PermGroFac,
        "Age": [x + birth_age for x in range(Agent.T_cycle)],
    }
)

raw_data = {
    "Age": Agent.history["t_age"].T.flatten() + birth_age - 1,
    "pLvl": Agent.history["pLvl"].T.flatten(),
    "PermShk": Agent.history["PermShk"].T.flatten(),
    "TranShk": Agent.history["TranShk"].T.flatten(),
}

Data = pd.DataFrame(raw_data)

# Create an individual id
Data["id"] = (Data["Age"].diff(1) < 0).cumsum()

Data["Y"] = Data.pLvl * Data.TranShk

# Find Et[Yt+1 - Yt]
Data = Data.join(exp_df.set_index("Age"), on="Age", how="left")
Data["ExpIncChange"] = Data["pLvl"] * (
    Data["PermGroFac"] * Data["exp_prod"] - Data["TranShk"]
)

Data["Y_change"] = Data.groupby("id")["Y"].diff(1)
```

The authors run an "index" version of the regression

$$E_t[\Delta Y_{i,t+1}] = \alpha + \beta\Delta Y_{i,t} + f_i + \epsilon_{i,t}$$

We now estimate an analogous regression in our simulated population.

In [5]:

```
from linearmodels.panel.model import PanelOLS
import statsmodels.api as sm

Data = Data.set_index(["id","Age"])

# Create the variables they actually use
Data["ExpBin"] = 0
Data.loc[Data["ExpIncChange"] > 0, "ExpBin"] = 1
Data.loc[Data["ExpIncChange"] < 0, "ExpBin"] = -1

Data["ChangeBin"] = 0
Data.loc[Data["Y_change"] > 0, "ChangeBin"] = 1
Data.loc[Data["Y_change"] < 0, "ChangeBin"] = -1

mod = PanelOLS(Data.ExpBin, sm.add_constant(Data.ChangeBin), entity_effects=True)
fe_res = mod.fit()
print(fe_res)
```

```
                          PanelOLS Estimation Summary
================================================================================
Dep. Variable:                 ExpBin   R-squared:                        0.1928
Estimator:                   PanelOLS   R-squared (Between):             -0.0671
No. Observations:              100000   R-squared (Within):               0.1928
Date:                Tue, Feb 22 2022   R-squared (Overall):              0.1863
Time:                        14:48:42   Log-likelihood                -1.282e+05
Cov. Estimator:            Unadjusted
                                        F-statistic:                   2.332e+04
Entities:                        2334   P-value                           0.0000
Avg Obs:                       42.845   Distribution:                 F(1,97665)
Min Obs:                       2.0000
Max Obs:                       47.000   F-statistic (robust):          2.332e+04
                                        P-value                           0.0000
Time periods:                      46   Distribution:                 F(1,97665)
Avg Obs:                       2173.9
Min Obs:                       1857.0
Max Obs:                       2339.0

                             Parameter Estimates
==============================================================================
            Parameter  Std. Err.     T-stat    P-value    Lower CI    Upper CI
------------------------------------------------------------------------------
const          0.1249     0.0028     44.751     0.0000      0.1195      0.1304
ChangeBin     -0.4336     0.0028    -152.72     0.0000     -0.4392     -0.4281
==============================================================================

F-test for Poolability: 1.2382
P-value: 0.0000
Distribution: F(2333,97665)

Included effects: Entity
```

The estimated $\hat{\beta}$ is negative because in usual life-cycle calibrations, transitory shocks are volatile enough that mean reversion of transitory fluctuations is a stronger force than persistent trends in income age-profiles.

However, with less volatile transitory shocks, the regression coefficient would be positive. We demonstrate this by shutting off transitory shocks, simulating another population of agents, and re-running the regression.

In [6]:

```
%%capture
params_no_transitory = copy(params)
params_no_transitory.update(
    {
        "TranShkStd": [0.0] * len(params["TranShkStd"])
    }
)

# Create agent
Agent_nt = IndShockConsumerType(**params_no_transitory)
Agent_nt.solve()
# Run the simulations
Agent_nt.initialize_sim()
Agent_nt.simulate()
```

In [7]:

```
exp = [
    calc_expectation(Agent_nt.IncShkDstn[i], func=lambda x: x[0] * x[1])
    for i in range(Agent_nt.T_cycle)
]
exp_df = pd.DataFrame(
    {
        "exp_prod": exp,
        "PermGroFac": Agent_nt.PermGroFac,
        "Age": [x + birth_age for x in range(Agent.T_cycle)],
    }
)

raw_data = {
    "Age": Agent_nt.history["t_age"].T.flatten() + birth_age - 1,
    "pLvl": Agent_nt.history["pLvl"].T.flatten(),
    "PermShk": Agent_nt.history["PermShk"].T.flatten(),
    "TranShk": Agent_nt.history["TranShk"].T.flatten(),
}

Data = pd.DataFrame(raw_data)

# Create an individual id
Data["id"] = (Data["Age"].diff(1) < 0).cumsum()

Data["Y"] = Data.pLvl * Data.TranShk

# Find Et[Yt+1 - Yt]
Data = Data.join(exp_df.set_index("Age"), on="Age", how="left")
Data["ExpIncChange"] = Data["pLvl"] * (
    Data["PermGroFac"] * Data["exp_prod"] - Data["TranShk"]
)

Data["Y_change"] = Data.groupby("id")["Y"].diff(1)
```

In [8]:

```
# Create variables
Data["ExpBin"] = 0
Data.loc[Data["ExpIncChange"] > 0, "ExpBin"] = 1
Data.loc[Data["ExpIncChange"] < 0, "ExpBin"] = -1

Data["ChangeBin"] = 0
Data.loc[Data["Y_change"] > 0, "ChangeBin"] = 1
Data.loc[Data["Y_change"] < 0, "ChangeBin"] = -1

Data = Data.set_index(["id","Age"])
mod = PanelOLS(Data.ExpBin, sm.add_constant(Data.ChangeBin), entity_effects=True)
fe_res = mod.fit()
print(fe_res)
```

```
                          PanelOLS Estimation Summary
================================================================================
Dep. Variable:                 ExpBin   R-squared:                        0.0072
Estimator:                   PanelOLS   R-squared (Between):          -4.494e-05
No. Observations:              100000   R-squared (Within):               0.0072
Date:                Tue, Feb 22 2022   R-squared (Overall):              0.0076
Time:                        14:48:46   Log-likelihood                -1.389e+05
Cov. Estimator:            Unadjusted
                                        F-statistic:                      710.03
Entities:                        2334   P-value                           0.0000
Avg Obs:                       42.845   Distribution:                 F(1,97665)
Min Obs:                       2.0000
Max Obs:                       47.000   F-statistic (robust):             710.03
                                        P-value                           0.0000
Time periods:                      46   Distribution:                 F(1,97665)
Avg Obs:                       2173.9
Min Obs:                       1857.0
Max Obs:                       2339.0

                             Parameter Estimates
==============================================================================
            Parameter  Std. Err.     T-stat    P-value    Lower CI    Upper CI
------------------------------------------------------------------------------
const          0.1083     0.0031     34.819     0.0000      0.1022      0.1144
ChangeBin      0.0848     0.0032     26.646     0.0000      0.0786      0.0911
==============================================================================

F-test for Poolability: 0.9781
P-value: 0.7692
Distribution: F(2333,97665)

Included effects: Entity
```

The estimated $\hat{\beta}$ when there are no transitory shocks is positive.