**Documentation:**

setup.jl

* Imports the necessary packages
* Includes the necessary scripts where different composite types and functions are defined.

master.jl:

* Initializes the fixed parameters, fp, (that will not change in the estimation or any other part of the script)
  + fp is of type fparams defined in params.jl.
  + initialization of fp is done by the function fparams in params.jl.
* Initializes moment structure, moments.
  + moments is of type structureM defined in params.jl
  + initialization of moments is done by the function initmom in params.jl
* Sets the guess parameter vector, initp0
* Calculates the moments from the data and creates the weight matrix for the objective function.
  + CalcDataBootMom! is the function that makes the calculations. It can be found in moments.jl.
* Calculates the objective function.
  + Objectivefunc! is defined in wrapper.jl.

wrapper.jl:

* Includes the main routine objectivefunc!() routine.
  + This routine first converts the initp0 array to params type.
    - Using pv2p0() defined in params.jl
  + Initializes gr, type grids that contains the parameters for constructing the grids for some of the variables. Defined in params.jl.
  + Initializes two instances of type single\_1 in params.jl using the function initEVEDU defined in params.jl.
  + Initializes sin, an instance of type structureM defined in params.jl, using the function initsim() defined in params.jl.
  + Draws the random components rvw and rvu.
  + Solves the model (the first for loop)
    - For each age (starting from the last one), value of experience grid points, value of assets grid points and value of previous labor supply choice, calls get\_cEmtr!() which is defined in solvesim.jl.
  + Simulates the life-cycle paths for each individual and each repetition, at each age (starting from the first age value) calls getopt!() which is defined in solvesim.jl.
  + Once the sim, lEV and lEDU are filled with the values calculated by get\_cEmtr!() and getopt!(), CalcSimMom!() calculates the moments (statistics) using sim.
  + Objective function is defined by the transpose of the difference between the data and simulated moments times a weight matrix (calculated in CalcDataBootMom! and stored as a field in moments) times the difference. The last two lines calculate that.

params.jl

* Defines type params and function params()
  + Type params is used to store the parameters that will be estimated.
  + Function params() initializes an instance of type params (not used in the code anymore – I was using this while testing the code.)
* Defines type fparams and function fparams()
  + Type fparams is used for storing the fixed parameters, settings of the model and simulation. We’ll also need to transfer this whole thing or parts of it for parallelization.
  + Function fparams() initializes an instance of type fparams.
* Defines type single\_1 and function initEVEDU()
  + Type single\_1 is used to store the values of the expected value function and the expected marginal utility functions at different values for the state variables (age, experience, last period’s labor supply choice). Both lEV and lEDU are used in the simulation so they are the main objects that we need to carry around or somehow let the workers have Access to during parallelization.
* Defines type sim\_t, OhCh\_t, Ch\_t and Oh\_t and function initsim()
  + Type sim\_t stores the simulated data. OhCh\_t is the subfield of sim\_t and Ch\_t and Oh\_t are subfields of OhCh\_t. Oh\_t refers to state variables, Ch\_t refers to choice variables.
  + sim which is an instance of type sim\_t is another main object for the code.
* Defines type structure and function initmom()
  + structureM holds information related to moments (both calculated from the data and the simulated data structure).
  + initmom() initializes an instance of structureM.
* Defines pv2p0()
  + This is a function to convert to initial guess parameter vector (Array{Float64,1}) to the params type.

grids.jl

* Defines the popgerr() function.
  + This function constructs the grids for the random components (the random shock in the utility function and the random shock in the wage function).
  + The doesn’t really include a random components. Only makes use of norminvcdf().
* Defines the function popge()
  + This function constructs the grids for the experience variable (which a continuous variable).
* Defines the function popgk()
  + This function constructs the grids for the assets variable, which is continuous.
* Defines the function nextwbds()
  + This function constructs the bounds for next year’s experience based on this period’s labor supply choice.
* Defines the function bounds()
  + This function determines the interval a value of a variable falls into, given the grid structure of the variable.
* Defines the function ilocate()
  + This function locates the bounds and thus the index (in the array) of a given value of a variable.
  + Used only for the random shocks.
* Defines the function trprob()
  + This function calculates the probabilities of a value of a random shock falling into each grid.

funcs.jl

* Includes LinInterp1d()
  + A function for linear interpolation along one dimension.
* Includes LinInterp2d()
  + A function for linear interpolation along 2 dimensions.
* Includes util()
  + This function defines the utility function of an individual.
* Includes dudc()
  + This function is the marginal utility of consumption (the derivative of the utility function with respect to consumption).
* Includes wage()
  + This function defines the wage equation.

moments.jl

* Includes CalcMom()
  + This is an intermediate function used to calculate the data moments.
  + This function is used by CalcDataBootMom!() defined in moments.jl.
* Includes CalcDataBootMom!()
  + This function changes the values stored in moments.
  + This function calculates the data moments (stores them in moments.dtamom) and the bootstrapped data moments to form the weight matrix (stores it in moments.wgtcov) which will both be used in the objectivefunc!() routine defined in wrapper.jl.
* Includes CalcSimMom!()
  + This function changes the values stored in moments.
  + This function calculates the moments from the simulated data (sim) and stores them in moments.simmom.
  + The moments.simmom will then be used in objectivefunc!() routine defined in wrapper.jl.

solvesim.jl

* Includes solveEE()
  + This is an intermediate function that solves the consumption decision of an individual.
  + This function will be used in the get\_cEmtr() function defined later in the solvesim.jl.
  + The function spits out the optimal level of assets for the next period and the closest grid points to that value.
* Includes get\_cEmtr!()
  + This is the main routine for the solution.
  + This function fills the lEV and lEDU structures.
  + If there will be any parallelization for the solution, it will happen here.
* Includes getopt()
  + This is the main routine for the simulation.
  + This function uses the lEV and lEDU structures to fill the sim structure.
  + The parallelization for the simulation part will happen here.