# humans.py

import: Agent from mesa.agent, college\_likelihood from fnnr\_config\_file, random, math, decimal

create lists: single\_male\_list, married\_male\_list, human\_birth\_list, human\_death\_list, human\_avoidance\_list (contains latest coordinates of human movement for monkeys to avoid), birth\_flag\_list, marriage\_flag\_list, human\_demographic\_structure\_list, hh\_migration\_flag (for head of household), num\_labor\_list (households are tracked by list indices), human\_marriage\_list, head\_of\_household\_list, former\_hoh\_list (for heads of households who have migrated), hh\_size\_list, total\_migration\_list (not cumulative; total at a time), total\_re\_migration\_list (cumulative), first\_step\_income\_list (to be used in the first step only)

Define \_readCSV, which reads in .csv files

class Human(Agent):

Set up all attributes (see model.py for the full list).

Each step, loop through all individual agents.

First step of the model only (if self.model.time == 1/73):

if not migrated (self.migration\_status ==0):

add self.income\_local\_off\_farm to household\_income\_list[self.hh\_id] (the household is the list index)

set first\_step\_income\_list[self.hh\_id] flag to 1 so duplicate incomes aren't added for each household member

set head of household for each household

calculate total number of laborers (num\_labor\_list[self.hh\_id] += 1 for each agent whose self.work\_status == 1)

calculate single males and married males in lists (single\_male\_list takes both self.unique\_id and self.hh\_id as arguments because married women migrate to their husband's household, so the husband's self.hh\_id needs to be stored explicitly)

elif migrated (self.migration\_status == 1):

calculate total number of migrants (set total\_migration\_list[self.hh\_id] +=1 for each agent)

define step() function (for each step of the model):

for non-migrants:

check education/college migration or retirement

check migration (1/73 chance per 5-day time-step, or on average, once a year)

check head of household

move humans on the grid

for migrants:

check re-migration (1/73 chance per 5-day time-step, or on average, once a year)

check marriage (1/73 chance per 5-day time-step, or on average, once a year)

check death (1/73 chance per 5-day time-step, or on average, once a year)

age the human agent

check age\_category

add 1/73 (in model years) to the last\_birth\_time

check birth possibility for married females of birth-giving age

check and shuffle single\_male\_list

define movement() function:

clear human\_avoidance\_list every step after it reaches a certain length (e.g. everyone has moved)

masterdict = self.model.saveLoad(masterdict archive from model.py)

if self.current\_position not in masterdict where it is out of monkey range (such as low elevation):

human\_avoidance\_list.append(self.current\_position)

also, human\_avoidance\_list.append(all 8-cell neighbors of self.current\_position)

if self.resource\_check == 0 (the human has not gathered the resource yet):

self.move\_to\_point(self.resource\_position)

if self.current\_position == self.resource\_position (if agent is at resource):

set self.resource\_check to 1

elif self.resource\_check == 1 (the human has gathered the resource):

self.move\_to\_point(self.home\_position)

Reset resource from random choice in list of resources (only one may be gathered at a time); from model import resource\_dict

resource = random.choice(resource\_dict[self.hh\_id])

set new resource and repeat gathering process

define age\_check() function (working status, education status, death rates, college/university migration):

if 15 < self.age < 59:

if self.work\_status == 0:

set self.work\_status to 1

set num\_labor\_list[self.hh\_id] += 1

else:

set self.work\_status to 0

if 7 < self.age < 19 and random.random() < 0.9:

self.education +=1

This means on average, FNNR residents will have about 10-11+ years of education; somewill have more, and some less.

Death rates per are are also calculated.

        if self.age <= 6:

            self.death\_rate = 0.00745 \* 0.5

        elif 6 < self.age <= 13:

        self.death\_rate = 0.0009 \* 0.5

        elif 13 < self.age <= 16:

            self.death\_rate = 0.00131 \* 0.5

        elif 16 < self.age <= 21:

            self.death\_rate = 0.00196 \* 0.5

        elif 21 < self.age <= 60:

            self.death\_rate = 0.001291 \* 0.5

        elif 60 < self.age:

            self.death\_rate = 0.05354 \* 0.5

0.5 is a changeable multiplier that depends on the frequency of death checks (in this case, yearly); what matters is that death rates are proportional by age category.

if 16 < self.age < 20 and random.random() < (0.0192 \* college\_likelihood) and self.migration\_status == 0 and random.random() < ((1/73) / 4):

Division by 4 in the odds represents the chance triggering once in four years during college-going age; college\_likelihood is set in fnnr\_config\_file.py

If these chances succeed, this human agent goes to college and is removed from the household

set self.migration\_status to 2

set self.education += 4 years

Out-migration/death-like attribute changes:

set hh\_size\_list[self.hh\_id] -= 1

set self.hh\_id to 'Migrated' and take care of head\_of\_household reassignment (if HoH)

set num\_labor\_list[self.hh\_id] -= 1 if applicable

human\_demographic\_structure\_list[self.age\_category] -= 1

remove human agent from schedule and grid

define check\_age\_category() function:

set age\_category to 1-9 if male and to 10-19 if female (checks as the person ages)

define hoh\_check() function:

If the head of the household has migrated, designates the next capable person as a substitute head of household; this slows down resource gathering temporarily

define birth\_check() function:

Add children to the reserve; this is triggered for married females of age only

if self.children < self.birth\_plan and if self.last\_birth\_time >= random.uniform(1, 4): (random.uniform means random float number)

determine attributes for children (gender is randomly generated with a 50/50 chance; see model.py for normal human generation rules) and add the new human agent to the schedule

ind = Human(for attribute listing, see model.py)

self.model.schedule.add(ind)

set self.model.number\_of\_humans += 1

set self.model.human\_id\_count += 1

                hh\_size\_list[self.hh\_id] += 1

                human\_birth\_list.append(last + 1)

                if ind.gender == 1, aka female:

                    human\_demographic\_structure\_list[0] += 1

                elif ind.gender == 2, aka male:

                    human\_demographic\_structure\_list[10] += 1

   define death\_check() function:

Small chance of dying every step; chance increases the older one gets, see age\_check()

        if random.random() < self.death\_rate:

remove self as head of household (if applicable) or former head of household if migrated, remove self from single\_male\_list or single\_married\_list (if applicable), subtract 1 from num\_labor\_list[self.hh\_id] if working, subtract 1 from hh\_size\_list

append self to human\_death\_list

            set self.model.number\_of\_humans -= 1

            set human\_demographic\_structure\_list[self.age\_category] -= 1

            self.model.schedule.remove(self)

            if self in self.model.grid:

                self.model.grid.remove\_agent(self)

define marriage\_check(self):

This function occurs approximately once a year (1/73) chance and thus uses a yearly marriage rate.

The below probability triggers for everyone:

if random.random() < 0.00767:

            marriage\_flag\_list.append(1)

The below only happens from the marrying female's point of view:

if random.uniform(20, 30) < int(self.age) < 45 and int(self.gender) == 2 and int(self.marriage) != marriage\_flag\_list != [] and self.migration\_status == 0:  (marriage occurs)

            set self.marriage = 1

            marriage\_flag\_list.remove(1)

modify household size list as female moves to male's household

add husband (randomly chosen from single male list) to married male list

human\_marriage\_list[self.hh\_id] += 1

define migration\_check() function:

        from land import non\_gtgp\_part\_list, gtgp\_part\_list, non\_gtgp\_area\_list

        self.non\_gtgp\_area = non\_gtgp\_area\_list[self.hh\_id]

calculate non\_gtgp\_land\_per\_labor as self.non\_gtgp\_area / num\_labor\_list[self.hh\_id]

determine if human is part of a GTGP household

Shuang sets remittances of the individual to the normal variate of (1200,400^2), but I think this is wrong because it can generate negative numbers; I chose to import mig\_remittances from his original data and have descendants inherit parents' attributes

The below is Shuang's migration formula, which is a lifetime probability (the yearly chance is divided by 45, for ages 15-59)

        prob = math.exp(2.07 - 0.00015 \* float(self.income\_local\_off\_farm) + 0.67 \* float(num\_labor\_list[self.hh\_id])

               + 4.36 \* float(self.migration\_network) - 0.58 \* float(non\_gtgp\_land\_per\_labor)

               + 0.27 \* float(self.gtgp\_part) - 0.13 \* float(self.age) + 0.07 \* float(self.gender)

               + 0.17 \* float(self.education) + 0.88 \* float(self.marriage) +

               1.39 \* float(self.work\_status) + 0.001 \* float(self.mig\_remittances))  # Shuang's formula

        mig\_prob = (prob / (prob + 1) / 45)

Finally, out-migration can only occur if there is at least one other person in the household.

        if random.random() < mig\_prob and hh\_size\_list[self.hh\_id] >= 2:

modify household size list, head of household status, self.hh\_id, self.migration\_status, num\_labor\_list if working, total migration list, self.work\_status, self.hh\_id

            add migrant remittances to household income

            if self in self.model.grid:

                self.model.grid.remove\_agent(self)

def re\_migration\_check(self):

        if self.migration\_status == 1:

            prob = math.exp(-1.2 + 0.06 \* float(self.age) - 0.08 \* self.mig\_years)

re\_mig\_prob = (prob / (prob + 1) / 45) # 45 for ages 15-59; migration is a lifetime, not yearly, probability

            self.mig\_years += 1

            if random.random() < re\_mig\_prob: they re-migrate

adjust household size list, num\_labor\_list if working, migration status, migration remittances on household income, household ID, number of years migrated, head of household if one was previously HoH

add self.unique\_id to total\_re\_migration\_list and remove from total\_migration\_list

define move\_to\_point()/move\_to() functions:

Moves human agent to assigned point according to resource-gathering frequency (higher frequency = 'exposed' to monkeys more in the wild while gathering resources)

For example, if a human agent gathers a resource 5 times a month and their resource is 21 grid coordinates away, they move approximately 4 grid coordinates towards the pixel each step.

This is not accurate in real time (humans move much faster than depicted), but makes it so that resources are gathered proportionally to each other.

The destination for a human at any given time is always either a resource or their home position (household coordinate).

class Resource(Agent):

set up position, freuency, hh\_id, type

see model.py; these are imported from 'hh added to the grid