model.py

import modules/libs: Model from mesa.model, MultiGrid from mesa.space, monkeys, environment, humans, land, family_setting/human_setting/run_setting from fnnr_config_file, pickle

set up lists: global family IDs, households # lists all household IDs from Shuang's data

load: vegetation ASCII .txt file (aggregated to ~85 x 100), elevation ASCII .txt file # aggregated to ~85 x 100, household buffer ASCII .txt file (default 400m; aggregated to ~85 x 100), farm buffer ASCII .txt file (default 300m; aggregated to ~85 x 100), PES buffer ASCII .txt file (default 200m; aggregated to ~85 x 100)
import forest buffer ASCII .txt file # default 200m; aggregated to ~85 x 100
*Note: I explain how to generate buffer files of different distances in FNNR-ABM User's Manual.docx, found on Github.

set up masterdict (vegetation types) and resource_dict (for human resource gathering)

class Movement: # this is the main model
initiate self ("model"), width, height, continuation/torus property, time (step), step within the current year, number of monkey families (automatically determined by fnnr_config_file setting), monkey birth count, monkey death count, monkey ID count, number of humans, model grid type (with or without humans, automatically determined by fnnr_config_file setting), model run type (first run to load new settings or normal run, automatically determined by fnnr_config_file setting), and human ID count

grid width = 85
grid height = 100
It is possible to change these numbers, but then many other attributes such as vegetation, elevation, and buffer ASCII files, as well as the parameters for how far each monkey travels per step, must also be changed accordingly or else the model will be warped/inaccurate; also, the model will run much more slowly if larger than 85 x 100

empty_masterdict = {'Outside_FNNR': [], 'Elevation_Out_of_Bound': [], 'Household': [], 'PES': [], 'Farm': [], 'Forest': [], 'Bamboo': [], 'Coniferous': [], 'Broadleaf': [], 'Mixed': [], 'Lichen': [], 'Deciduous': [], 'Shrublands': [], 'Clouds': [], 'Farmland': []}
These include all of the vegetation types monkeys can step on

# generate land
if model run type == 'first_run':
    self._populate()
    The coordinates that make up each vegetation category will populate empty_masterdict.

elif model run type == 'normal_run':
    empty_masterdict = self.saveLoad()

empty_masterdict is opened from a saved record instead of being populated by an imported file.

Generate Resources -
if self.grid_type == 'with_humans':
for line in _readCSV('hh_survey.csv'): # hh_survey.csv contains adapted resource data from Shuang's original data file
resource = Resource(_readCSV('hh_survey.csv')) # create agent and add to grid
resource_dict.setdefault(hh_id_math,
self.grid.place_agent(resource) # adds resource points to the landscape
The following are the types of resources the humans can gather: fuelwood, herbs, bamboo, mushrooms, fodder, fish, and other.

Generate Land Parcel Agents -
for line in _readCSV('hh_land.csv): # hh_land.csv contains adapted GTGP/land area data from Shuang's original data file
for i in range(1,6): $ up to 5 land parcels per household
non_gtgp_area = float(total_rice) + float(total_dry) - float(gtgp_dry) - float(gtgp_rice) # calculate non_gtgp_area and gtgp_area; the rest is imported
gtgp_area = float(gtgp_dry) + float(gtgp_rice)

lp = Land(household ID, GTGP enrollment status, original head of household's age/gender/education, household size, total rice crop, GTGP rice crop, total dry crop, GTGP dry crop, land type (rice or dry), land travel time, plant type (specific crop), non-GTGP output, pre-GTGP output, household size, calculated non-GTGP area, calculated GTGP area)
self.schedule.add(lp)

Repeat for both non-GTGP and GTGP land parcel (since they import different parts of ‘hh_land.csv’)

Generate Human Agents -
Set self.number_of_humans and self.human_id_count = 0 # counters that will be increased later
for line in _readCSV(‘hh_citizens.csv’):
starting_position = _readCSV(‘household.csv’)
resource = random.choice(resource_dict[hh_id]) # sets resource that person gathers
age_category = 1-9 if male depending on age, or 10-19 if female depending on age
if gender == 2 (female):
if marriage == 1 (married) and age < 45:
children = random.randint(0, 4) # existing amount of children so births aren’t inflated
birth_plan_chance = random.random()
if birth_plan_chance < 0.03125:
birth_plan = 0
elif 0.03125 <= birth_plan_chance < 0.1875:
birth_plan = 1
elif 0.1875 <= birth_plan_chance < 0.5:

birth_plan = 2
elif 0.5 <= birth_plan_chance < 0.8125:
birth_plan = 3
elif 0.8125 <= birth_plan_chance < 0.96875:
birth_plan = 4
else:
birth_plan = 5
elif gender != 2 (male):
birth_plan = 0
last_birth_time = random.uniform(0, 1) # numbers vary
human_demographic_structure_list[age_category] += 1 for that person
filter out all humans with a -3 as a value anywhere in Shuang's data (doesn't exist)
self.number_of_humans += 1
self.human_id_count += 1 # unlike number_of_humans, human_id_count never goes down
human = Human(calculated starting position, household ID, age, resource gathering flag, starting position on grid, chosen resource position on grid, frequency of resource gathering, gender, education, work status, marriage status, past household ID, number of years migrated, migration status, GTGP participation, non-GTGP area, migration network (binary 0 or 1), migration remittances, local off-farm income, last birth time, death rate (set later), age category, number of existing children, birth plan (total number of children))
self.schedule.add(human)
self.grid.place_agent(human, starting_position)
# note how agents that change are added to the schedule, and agents that show up in the visualization are added to the grid; in other words, land parcel agents are only added to the schedule, resource agents are only added to the grid, and human and monkey agents are added to both

Note that migrant humans (each household starts with 0 or 1 migrants) are added separately: they have the same attributes as a human in the villages and behave the same once they re-migrate, but their attributes import from different places in 'hh_citizens.csv'

Generate monkey family agents below -
for i in range(self.number_of_families): # this is set in fnnr_config_file.py
starting_position = random.choice(startinglist)
saved_position = starting_position # preserve previous position while moving
from families import Family
family_size = random.randint(25, 45) # sets family size for each group--random integer
family_id = i
list_of_family_members = [] # will populate empty list later
family_type = 'traditional' # as opposed to an all-male subgroup
split_flag = 0 # binary: 1 means its members start migrating out to a new family
family = Family(Family ID, model, starting position, family size, list of family members, family type—traditional or all-male, saved position, and split_flag—a family group splits into two groups once it exceeds 45 members)

self.grid.place_agent(family, starting_position)
self.schedule.add(family)
global_family_id_list.append(family_id)

The below describes general individual monkey agents; note that this is nested within family generation
for monkey_family_member in range(family_size):
id = self.monkey_id_count
gender = random.randint(0, 1)
if gender == 1: # gender = 1 is female, gender = 0 is male.
# This is different than with humans (1 or 2)
female_list.append(id)
last_birth_interval = random.uniform(0, 2)
else:
male_maingroup_list.append(id) # as opposed to the all-male subgroup
last_birth_interval = -9999 # males will never give birth
mother = 0 # no parent check for first generation

The next part describes the ratio by age categories:

Group	Baby	Juvenile	Young	Sub-adult	Adult	Senior
Age thresholds	<1 year	1 to <3 years	3 to <7	7 to <10	10 to <25	25 or above
%	11%	16%	15%	20%	34%	4%
F:M ratio at each age group	1:1	1:1	1:1	1:1	2.9:1	All females

# starting representation of male defection/gender ratio
structure_convert = random.random()
if gender == 0 (male) and age_category == 4 (aged 10-25):
if structure_convert < 0.6:
set gender = 1 (convert some males to females to keep the 2.9:1 ratio)
set last_birth_interval and add to reproductive_female_list
Next, create the monkey:
monkey = Monkey(ID, model, gender, age, age category, family ID, last birth interval in years, mother ID)
self.number_of_monkeys += 1
self.monkey_id_count += 1
list_of_family_members.append(monkey.unique_id)
self.schedule.add(monkey)

def step(self):
# necessary; tells model to move forward every step
self.time += (1/73)
self.step_in_year += 1
if self.step_in_year > 73:

```
        self.step_in_year = 1  # start new year
    self.schedule.step()
```

def _readASCII() function, which reads in the imported vegetation/elevation/buffer .txt files

define _populate() function, which places vegetation tiles on the model grid

define saveLoad() function, which pickles objects so they can be loaded from a file instead of re-generated every model run