# Melodie: Agent-based Modeling in Python

21 January 2023

## Introduction

Agent-based models (ABMs) characterize physical, biological, and social economic systems as dynamic interactions among agents from a bottom-up perspective. The agents can be molecules, animals, or human beings. The interactions can be water molecules forming a vortex, ants searching for food, or people trading stocks in the market.

Agents' interactions can bring emergent properties to a system and turn it into a complex system. To model such mechanism is usually the core reason for using ABMs. Besides, taking social economic systems as example, ABMs are also flexible to consider agents' (1) heterogeneity (e.g., wealth, risk attitude, preference, decision-making rule, etc.) based on micro-data; (2) bounded rationality and adaptation behavior based on psychological and behavioral studies.

`Melodie` is a general framework for developing agent-based models (ABMs) in Python. It is published and maintained on the GitHub organization page of `ABM4ALL`[1], a developing community among agent-based modelers for sharing ideas and resources. Together with the code repository[2], we also published the documentation[3] of `Melodie`, including a tutorial explaining how a minimum example - an agent-based covid contagion model - can be developed with `Melodie` step by step.

## Statement of need

In the Python community, `Mesa` [@Mesa][4] and `AgentPy` [@AgentPy][5] are the two open-source frameworks for agent-based modeling. Following the tradition of `NetLogo` [@Netlogo][6], they both support interactive simulation but with different focus and style.

[1] Link to `ABM4ALL`: https://github.com/ABM4ALL.
[2] Link to `Melodie` repository: https://github.com/ABM4ALL/Melodie.
[3] Link to `Melodie` documentation: https://abm4all.github.io/Melodie/html/index.html.
[4] Link to `Mesa`: https://github.com/projectmesa/mesa.
[5] Link to `AgentPy`: https://github.com/JoelForamitti/agentpy.
[6] Link to `NetLogo`: https://ccl.northwestern.edu/netlogo/.

In summary, Melodie distinguishes from them in the four following aspects.

- First, Melodie separates an environment component from the model in Mesa and AgentPy for two dedicated tasks: (1) storing the macro-level variables; (2) coordinating the agents' decision-making and interaction processes.
- Second, Melodie enhances the data_collector component with higher configurability.
- Third, Melodie has a wider infrastructure coverage and provides two dedicated modules for scenario management, i.e., importing input data and deliver them to the model and its components.
- Fourth, Melodie includes two modules that are not provided in Mesa and AgentPy: Calibrator, and Trainer. With these two modules, Melodie supports (1) automatic calibration of scenario parameters, and (2) evolutionary training of agents.

# Overview

The modules in the Melodie framework can be organized into four clusters: Model, Scenario, Modeling Manager, and Infrastructure.

## Model

The modules in the Model Cluster focus on describing the target system. Developed with Melodie, a model object can contain following components:

- agent - makes decisions, interacts with others, and stores the micro-level variables.
- agents - contains a list of agents and provides relevant functions.
- environment - coordinates the agents' decision-making and interaction processes and stores the macro-level variables.
- data_collector - collects the micro- and macro-level variables from the agents and environment, and then saves them to the database.
- grid - constructed with spot objects, describes the grid (*if exists*) that the agents walk on, stores grid variables, and provides the relevant functions.
- network - constructed with edge objects, describes the network (*if exists*) that links the agents, and provides the relevant functions.

## Scenario

The modules in the Scenario Cluster focus on formatting, importing, and delivering the input data to the model, including

- DataFrameInfo and MatrixInfo - used to create standard data object for input tables.
- data_loader - loads all the input data into the model.

- `scenario` - contains all the input data that is needed to run the model, and can be accessed by the `model` and its components.

## Modelling Manager

To combine everything and finally start running, the Modelling Manager Cluster includes three modules, which can be constructed and run for different objectives:

- `Simulator` - simulates the logics written in the `model`.
- `Calibrator` - calibrates the parameters of the `scenario` by minimizing the distance between model output and empirical evidence.
- `Trainer` - trains the `agents` to update their behavioral parameters for higher payoff.

Both of `Calibrator` and `Trainer` modules are based on the genetic algorithm (GA), and the `Trainer` framework is introduced in detail in [@Yu].

Taking the covid contagion model in the tutorial as example, as shown below, the `simulator` is initialized with a `config` object (incl. project name and folder paths) and the class variables of the `model`, the `scenario`, and the `data_loader`.

```
from Melodie import Simulator
from config import config
from source.model import CovidModel
from source.scenario import CovidScenario
from source.data_loader import CovidDataLoader

simulator = Simulator(
     config=config,
     model_cls=CovidModel,
     scenario_cls=CovidScenario,
     data_loader_cls=CovidDataLoader
)
simulator.run()
```

At last, by calling the `simulator.run` function, the simulation starts.

## Infrastructure

The last Infrastructure Cluster includes the modules that provide support for the modules above.

- `Visualizer` - provides the APIs to interact with `MelodieStudio` for visualization.
- `MelodieStudio` - another library in parallel with `Melodie`, which supports results visualization and interactive simulation in the browser.
- `Config` - provides the channel to define project information, e.g., project name, folder paths.

- `DBConn` - provides the functions to write to or read from the database.
- `MelodieException` - provides the pre-defined exceptions in `Melodie` to support debugging.

## Resources

On our GitHub organization page `ABM4ALL`, apart from the `Melodie` package and its documentation, we also published a series of example models showing how different modules can be used, including `Grid`, `Network`, `Calibrator`, `Trainer`, `Visualizer`, and `MelodieStudio`. These example models are also documented in the "Model Gallery" section in the `Melodie` documentation. Finally, for those who are familiar with `Mesa` or `AgentPy`, a comparison between `Melodie` and the two packages is provided in the documentation, based on the same covid contagion model developed with all the three packages.

## Acknowledgements

This work is not supported by any funding. Dr. Songmin Yu would like to thank the free and creative working atmosphere at Fraunhofer ISI, especially the inspiring talks and nice beer time with the colleagues. Zhanyi Hou would like to thank his supervisor, Prof. Shunkun Yang, and his research partners from Beihang University for their support and guidance for programming.

## References