Agent-Based Modeling: Week 3

Winter 2023

Jean Clipperton

Agenda:

- ODD + D framework
- Concept: Emergence
- Visualization:
    - Server file
    - Elements
- Model: Flocking

Grimm and Volker: ODD + D

- Puzzle / question
- Theory / reasoning
- Framework
- Do I buy it / so what?

GV: ODD + D Framework

Overview	**Purpose**
	State variables and scales
	Process overview and scheduling
Design Concepts	Design concepts
Details	Implementation details
	Initialization
	Input
	Submodels

GV: ODD + D Overview

- **Purpose** Who is the model for? (e.g. scientists, students / teachers, stakeholders, decision makers?)
- **Entities, state variables, and scales**: What kinds of entties are in the model, what state variables an dparameters do they have?
- **Process overview and scheduling** Describe scheduling, names for the processes, how the update process works.

GV: Design Concepts

Here, give an overview of all the pieces that come together to create the model. For example, you will want to discuss What the agents are, how they interact with their environments, any learning they are able to (particularly w/r/t their environment), and what random processes exist within the model.

GV: Details

This is where you provide the technical details of the model. Think of it as the actual recipe: **can someone read this component and recreate the model themselves?** *If the answer is no, you need to provide addtional detail*

Concept: Emergence

We'll go over textbook from Volker and Grimm -- as discussed, focusing on the key conceptual portions.

Emergence is the complex, often unexpected dynamics at both the individual and system levels that comes out of the underlying processes.

- Not the sum of the properiteis of the model's pieces
- Different type of result than indiviual-level properites or deciions
- It cannot be eawily predict from the properties of the individuals

Concept: Emergence

This is one of the key components that makes ABMs so useful: sometimes there are surprising results that emerge.

Common characteristics of emergent dynamics are that the results are interesting and hard to describe quantitatively and secondly there is some kind of 'warm up' or 'burn in' period before you observe the outcomes.

Visualization: How do you want to represent your model?

We had some experience with this last week and we're going to think about it a bit more formally now.

Focus on how you want the user to interact with your model *Later, we'll talk about exporting data but that will also factor in*

Plan ahead!

Think about what you are hoping to use the model for, what you want to communicate, and what pieces you'll need.

LIFE IS EASIER WHEN YOU HAVE A PLAN

One last consideration: Batch and Single runs

In our class, we've so far been doing what's called a *'single run'* vs a *'batch run'*. We'll dive into this more, but essentially a single run is just what it sounds like -- one run of the model.

When we move to exporting data and doing different sorts of analyses on the model, we'll transition to batch runs. In the batch run scenario, the visualizations will play a small role but **the parameters you allow the user to set will still factor prominently.**

Server file

Relevant elements:

- Header: what we import
- Agent portrayal
- Any reporting elements to display
- Canvas / grid
- Elements in the vizualization
- Set up the server

Server File: Header

As you know, there are different ways to import the visualization. I erred on the side of the most information. Below, you'll see that our imports allow us to portray the grid (CanvasGrid), render the model (ModularServer), make any reporting elements (ChartModule,TextElement), and call our model (here, this is an example from Schelling).

```
from mesa.visualization.modules import CanvasGrid
from mesa.visualization.ModularVisualization import ModularServer
from mesa.visualization.modules import ChartModule, TextElement
from mesa.visualization.UserParam import UserSettableParameter

from model import SegModel
```

Server File: Agent portrayal

Here you will set up agent portrayal via draw function. Consider how you want the grid to appear (do you want the blank spaces to appear any way, for example?)

Next, consider how agents will be portrayed:

- Shape: will they be round, square, filled, open?
- Type: will you vary appearance by type? How so?

Server File: Any reporting elements to display

These are text elements that containg values for your trackers or reporting elements that you want to include. These may be percentages that are challenging to read / calculate from just looking at the gui or more specific / complicated calculations.

For example:

- You might display the count of agents if your model allows for death / removal / changing of types.
- You might want to know the ratio of some parameter value
- You may have a particular index or statistic that you want to calculate

Server File: Text element psuedocode

```
class name(TextElement):

    def render(self, model): #probably going to be model
        return "text and any variables  / calculations"
```

Server File: Text element example

For this, we're calculating and displaying the proportion of happy agents. Depending on how we want to do things and how we want to use them, we might calcuate these as variables to be used in other reporting features. It just depends.

```
class HappyElement(TextElement):

    def render(self, model):
        return "% Happy agents: " + str(round(
            (model.happy / model.num_agents) * 100, 1)) + "%"
```

Server File: Gui setup!

In this component, you'll set up the visualization in preparation for the next step. You will set up the model parameters. There are a ton of ways you might choose to do this -- I like to pull things together from earlier so that everything is tidy.

```
elements = NamedElement()
canvas = CanvasGrid(draw function witih width and pixels)
chart = ChartModule(label and name)
```

Server File: Gui setup!

In this component, we'll think about how the actual GUI will appear -- to what extent do we want user inputs (we dive more deeply into the mechanics soon), what options do we want?

We effectively are generating a dictionary of everything we want to include

```
height: Model.height #I like to call it this way vs restating
width: Model.height
user_inputs: (see next few slides)
```

Server File: graphics with the vizualization (call the server)

Here, we put it all together -- you'll call the modular server which will render our model.

```
server = ModularServer(
    Model,
    [canvas_element,text elements,
     charts ],
    "title",
    model_params
)
```

Elements of design

We're going to revisit the same elements we just discussed, but now with an eye toward design.

- Grid: how the elements appear
- User-inputs: parameter values the user can set
- Reporting elements: displayed statistics
- Graphics: in-the-moment graphs

We'll take these each in turn, but consider these factors when you're setting up your model.

Documentation

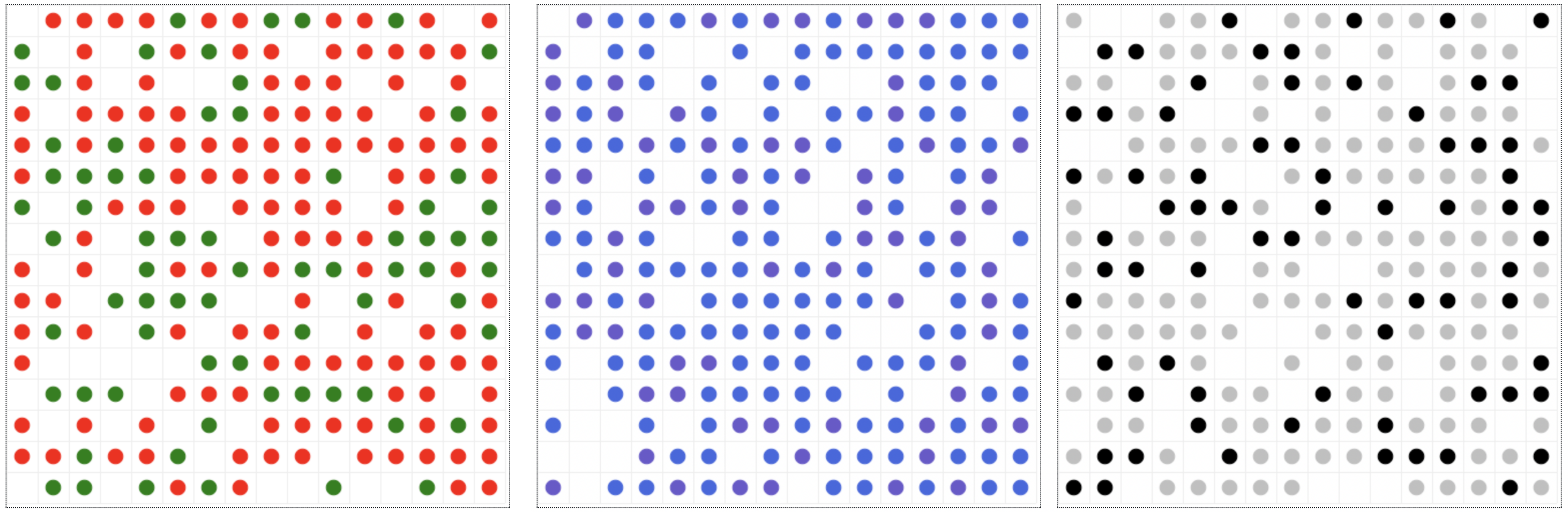
For mor information and detail, you can find Mesa's documentaion on this here:

Mesa Documentation

Note that it's not always the most user friendly in how it's described or displayed.

Grid

The grid is how the actual model appears. Consider the dimensions you want for your visualization and how you want it to appear. Colors are important -- you want high contrast and to avoid red / green for colorblindness.



See this link for color names in Python.

User inputs: UserSettableParameter

We talked in the first week about how you want your user to interact with the model / gui. This is your opportunity to offer that.

Parameter types include:

‘number’ - a simple numerical input

‘checkbox’ - boolean checkbox

‘choice’ - String-based dropdown input, for selecting choices within a model

‘slider’ - A number-based slider input with settable increment

‘static_text’ - A non-input textbox for displaying model info.

User inputs: UserSettableParameter

My thoughts

Parameter types include:

‘number’ - I'm not sure this would be a good choice -- a slider gives you a constrained range

‘checkbox’ - this cold be interesting to turn on / off a feature

‘choice’ - I would turn this on with caution

‘slider’ - This is intuitive and easy to use for your user.

‘static_text’ - I don't think this would be a good choice vs other options.

Data collection and reporting

We're not getting into this just yet because it can be a bit complicated. We'll discuss a basic overview and return to this next week!

Server file

Relevant elements:

- Header: what we import
- Agent portrayal
- Any reporting elements to display
- Canvas / grid
- Elements in the vizualization
- Set up the server

How nice should it be?

This is a tough question to ask and depends on the following components:

1. Your own preferences (some people want things to be always polished)
2. The audience for the model (will the model be used by anyone other than you, for example)
3. The kind of project (is this a proof of concept approach, is this going to be how you generate findings, etc)

Model application: Flocking

Wilensky, U. (1998). NetLogo Flocking model. http://ccl.northwestern.edu/netlogo/models/Flocking. Center for Connected Learning and Computer-Based Modeling, Northwestern University, Evanston, IL.