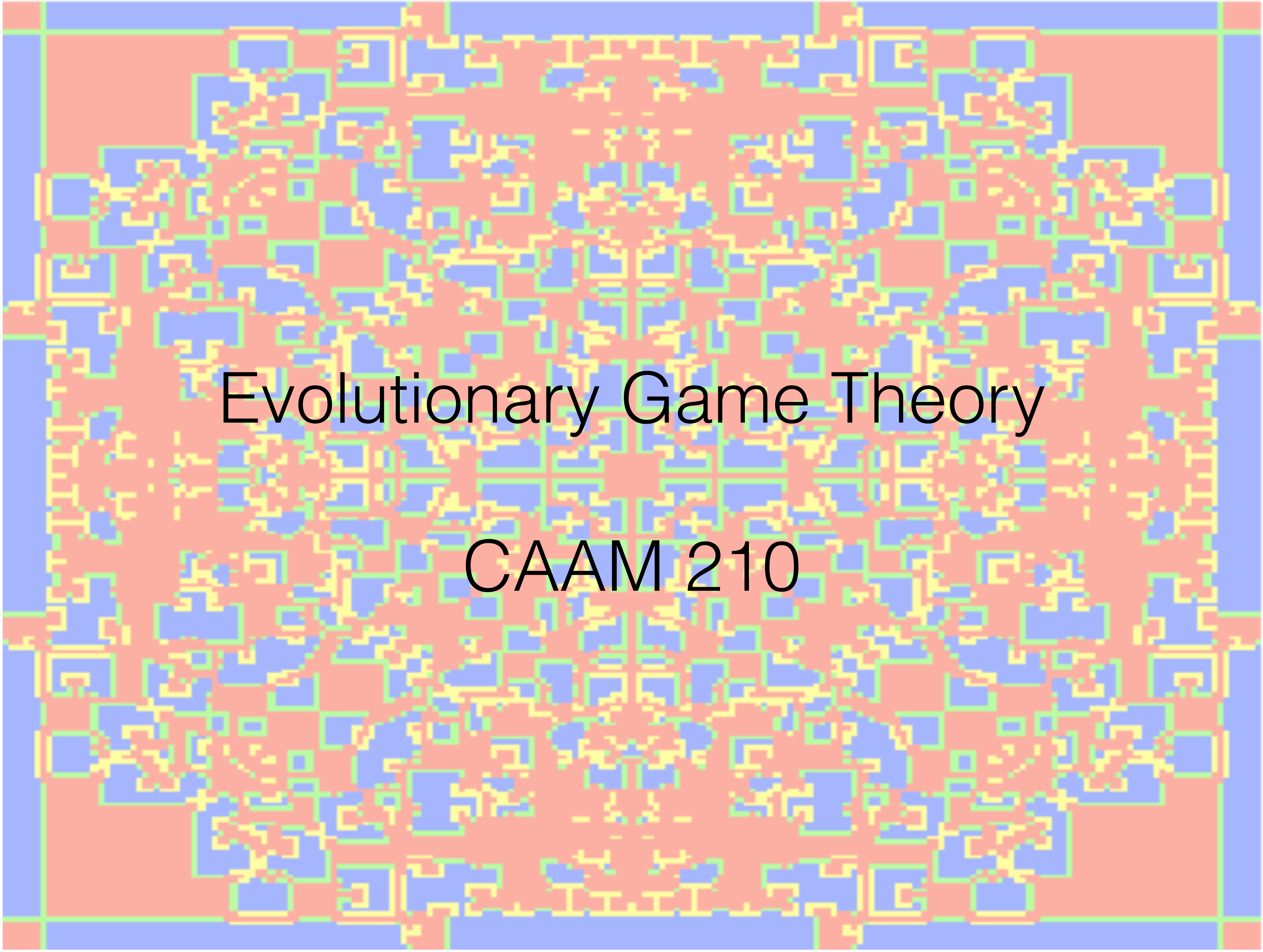


Evolutionary Game Theory

CAAM 210

Outline

function evodriver

- runs evo for values specified in notes

function evo(M,N,b,gen)

- calls score, advance, evodisp to play game and display colored matrices

- creates and plots fraction of cooperators plot

function S = score(A, b)

- called at each iteration, calculates score of each player

function An = advance(S, A)

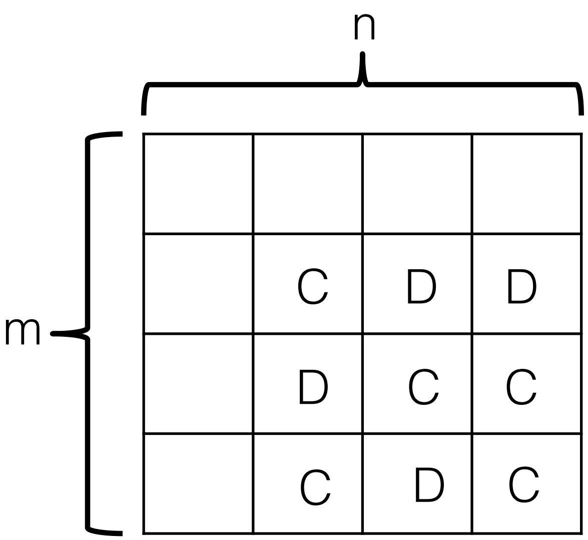
- called at each iteration, for every player finds the neighbor with the highest score, and changes the player's identity to the winning neighbor's identity

function evodisp(A, An)

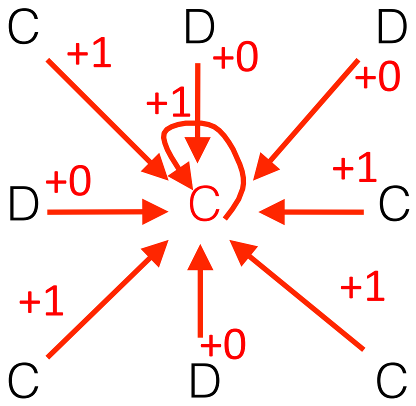
- creates color "slice" matrix for display

Scoring

C vs. C	->	each receives 1
C vs. D	->	C receives 0 and D receives *b* (b > 1, b = 1.9)
D vs. D	->	each receives 0



	C	D	D
	D	C	C
	C	D	C



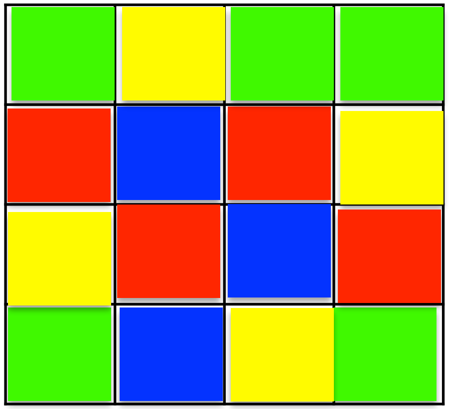
C scores 5 this round

- At each round, every player’s score is calculated in this manner.
- Each player also plays themselves.
- Importantly, the players on the *edges* must play those on the opposite edges, as if the game board were wrapped unto itself so all of the edges met.

Advancing

- After the score is calculated at each round, each player takes the identity of the neighbor (including themselves) with the *highest score* on the last round. If the highest scorer was a defector, the player under consideration becomes a defector, etc.
- The grid is *colored* to represent the change in identity of each player at each iteration, as follows:

🟦	C remains C
🟥	D remains D
🟨	C becomes D
🟩	D becomes C



fine print: this is just an example. Don't try to figure out the logic behind it – there isn't any.

How to color?

- using image on a matrix, for example **image(M)** will produce a colored plot such as those found in the notes.
- RGB triples can be used to encode the colors.
- The matrix has 3 “slices”. In other words, it is a “stack” of the m x n matrices you have worked with in this class to date. Each slice encodes one of the R, G, or B values.

🟦	RGB: 0 0 1
🟥	RGB: 1 0 0
🟨	RGB: 1 1 0
🟩	RGB: 0 1 0

```
>> M = rand(50,50,3);
>> image(M)
```