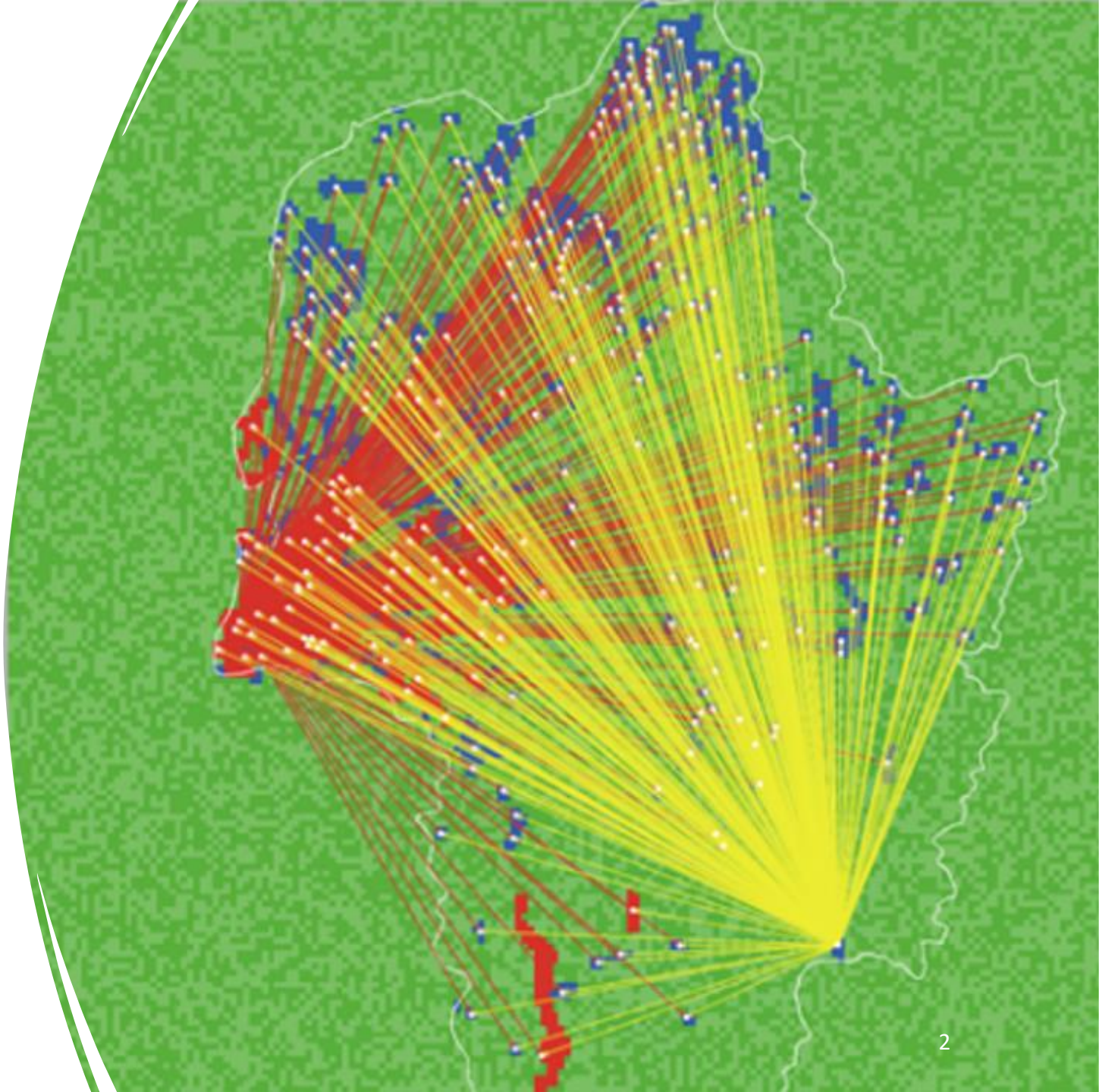
DESIGN CONCEPTS AND PROGRAMMING TIPS

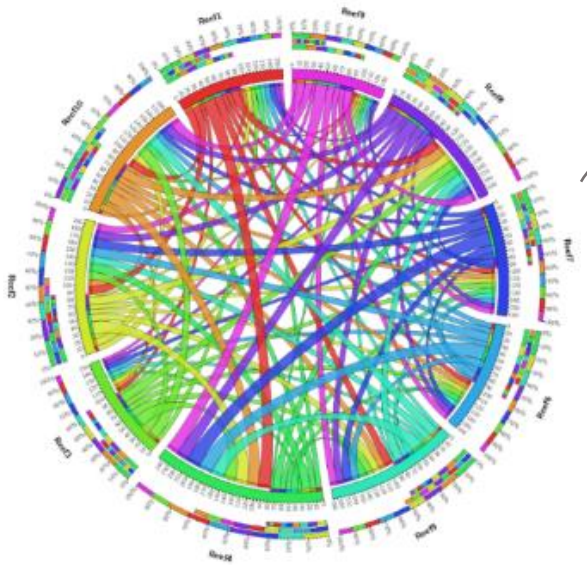
MODELING!

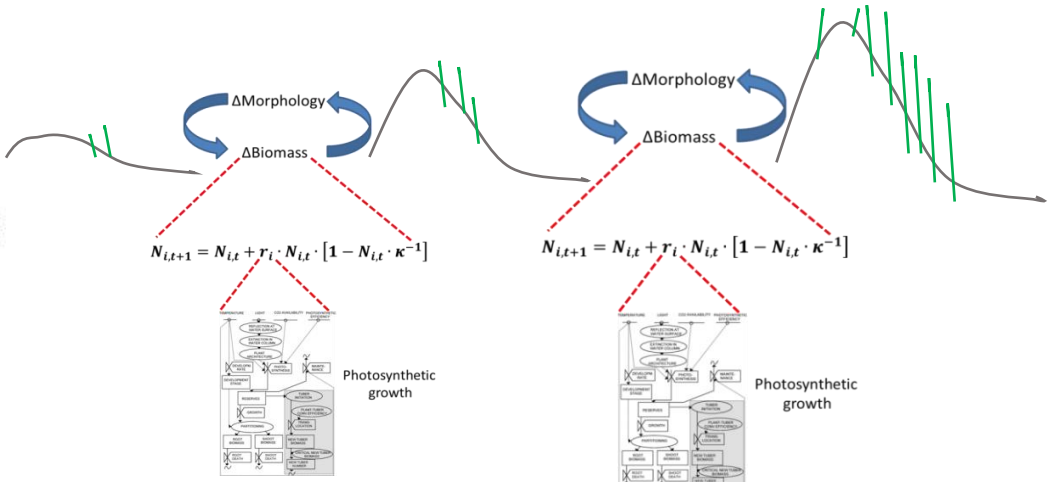
FW 508, Fall 2022

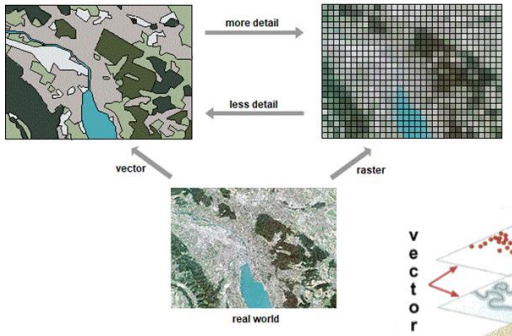
OVERVIEW

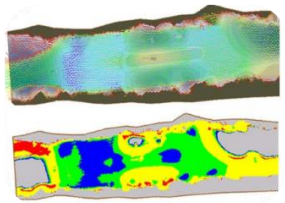
- **DESIGN**
  - Emergence
  - Sensing
  - Interactions
- **PROGRAMMING TIPS**

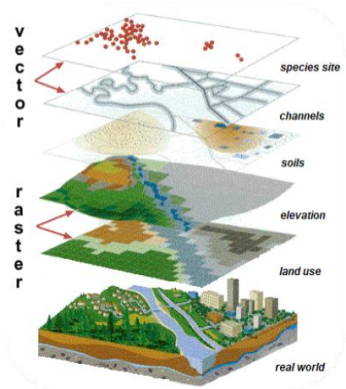


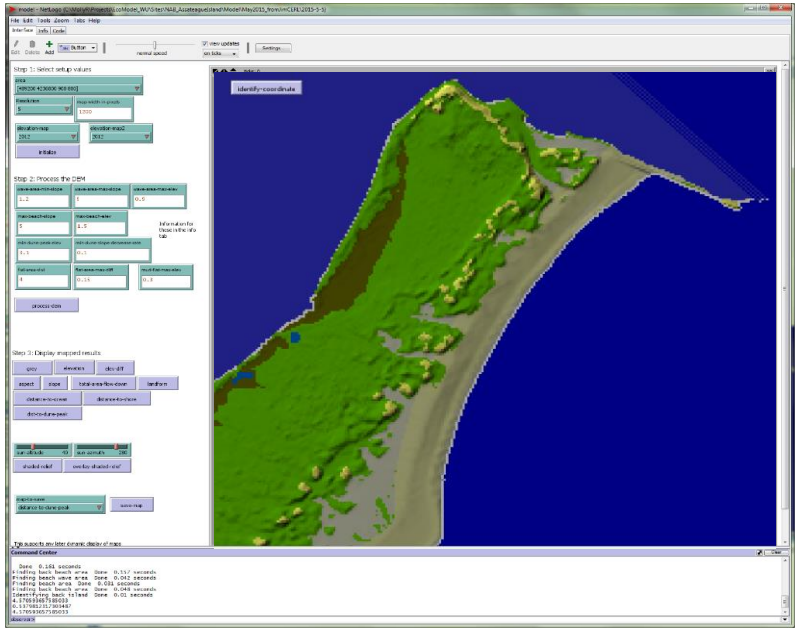












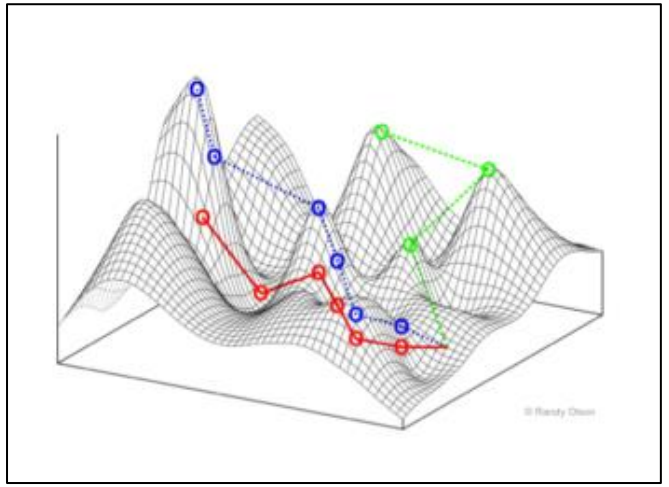
DESIGN

SETTING UP YOUR MODEL

Model design requires careful thinking about how agents interact, sense each other, and the potential emergent properties

Design concepts should be documented prior to building model (conceptual models, pseudocode, etc).

EMERGENCE



What is an emergent property?

- More than the sum of individual properties
- Different type of results than individual properties
- Cannot easily be predicted from interactions

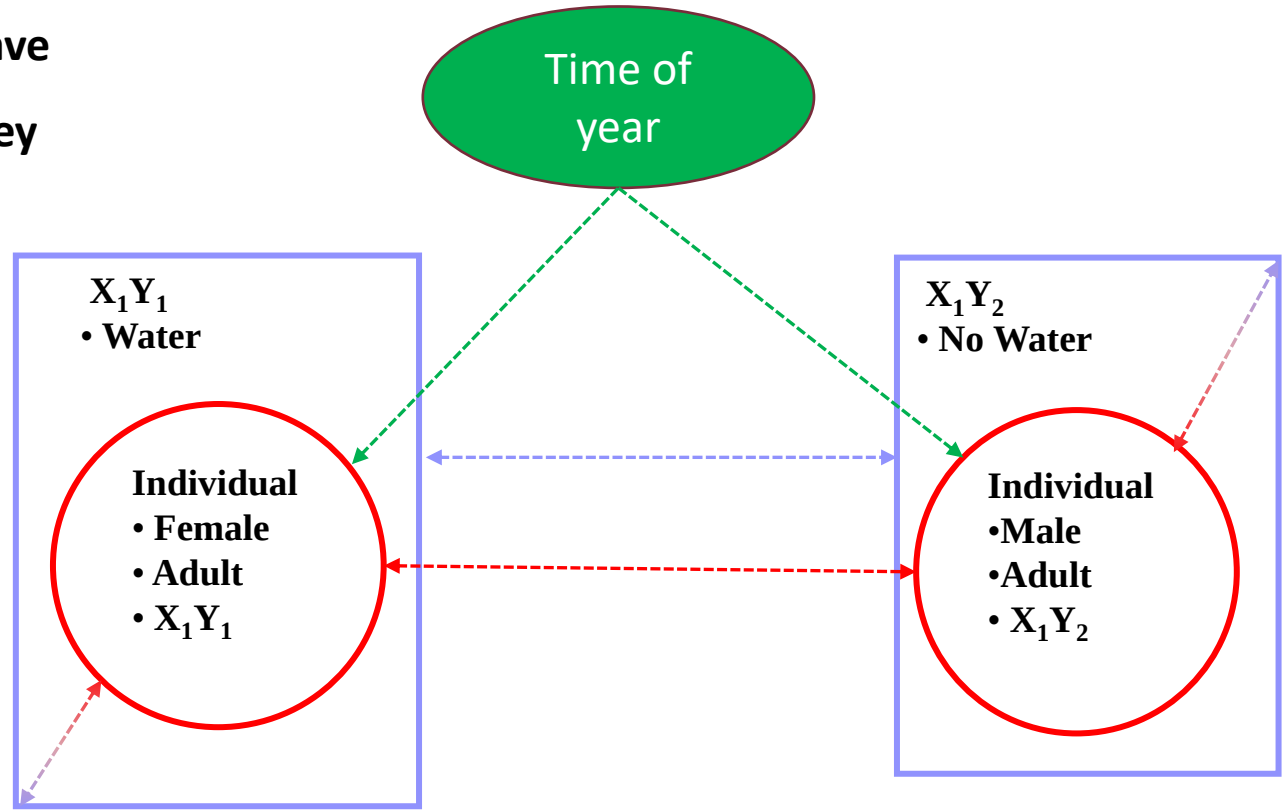
- https://www.youtube.com/watch?v=CipO1XGh4QM

SENSING & INTERACTION

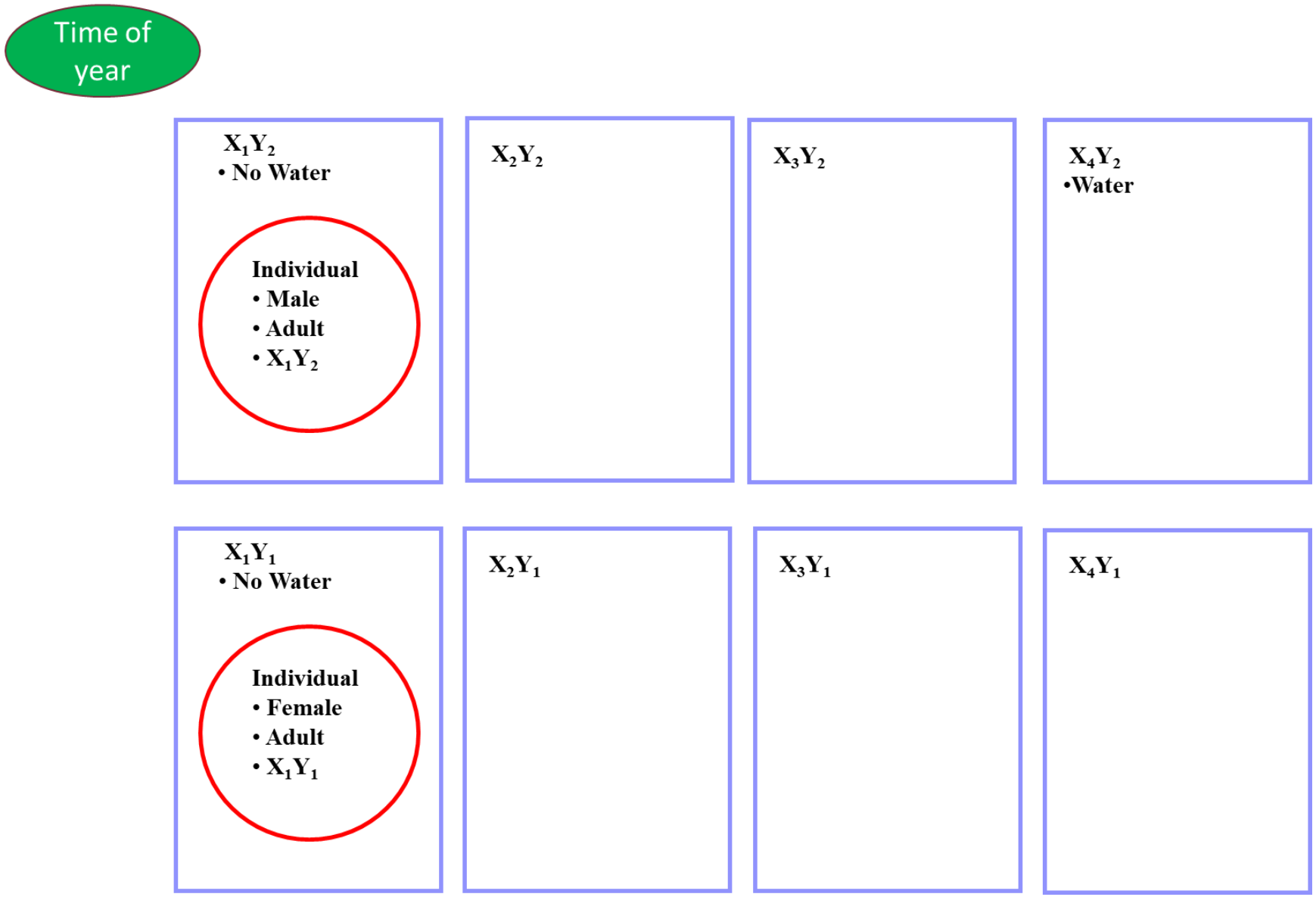
Gathering information

- How is information obtained by agents?
  - Do they just know it? Have to gather it?
  - Is it a property of other agents, patches, or global environment?
  - **Thought exercise: Assume these agents needed water at a certain time of year to breed. What would they have to sense? What are the interactions? How would they get information about time of year? What are the global variables? Patch? Turtle?**

Code (Show model 'Class example'):



WHAT ABOUT THIS?



Code:

HEROES AND COWARDS

I AIN'T SCARED!

In a group of people, each picks a *friend* and an *enemy*. In the first phase of the game, people act like cowards and move to keep their friend between them and their enemy. In the second phase, people act as heroes and move between their friends and enemies.

WHAT WILL HAPPEN?

SIMPLE RULES CAN LEAD TO COMPLEX PATTERNS

Keep it simple

- ABMs often start as series of if statements
- Start simple, add complexity once simple version works
- Write pseudocode, draw things out, etc
- Trial and error!

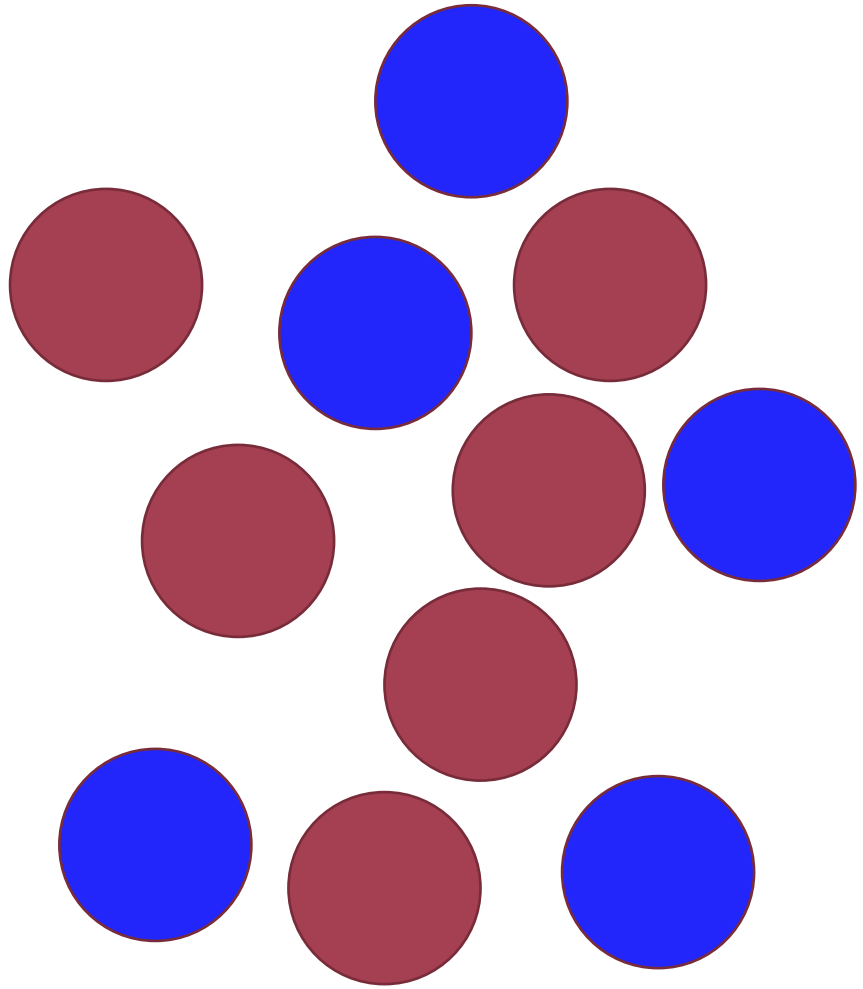
CODING IN NETLOGO

LESSONS LEARNED

NETLOGO CODING TRICKS

PRACTICE!

- Create *agentsets* and *lists* when needed to optimize loops
  - Agentsets
    - Let foo turtles with [color = “blue”]
      - Ask foo [commands]
    - Processed in random order
  - Lists
    - Let temp list (random 10) (random 9) (“test”)
      - Show list [10 9 “test”]
    - Processed in order

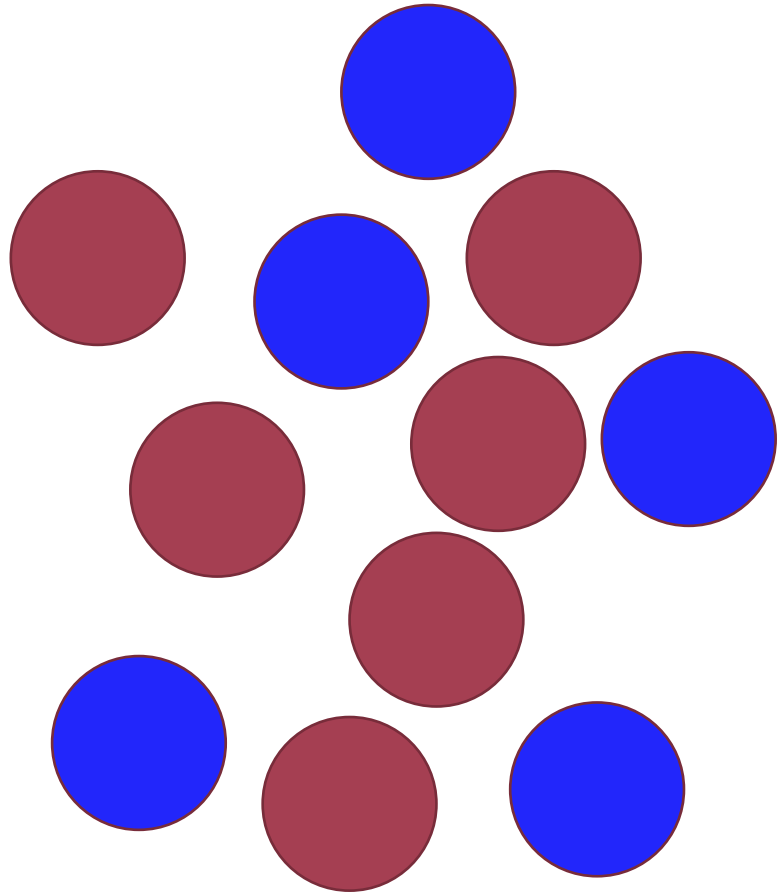


NETLOGO CODING TRICKS

PRACTICE!

- All netlogo agentsets and lists start with 0!

- Use *show* and *print* to verify code
  - Show count foo
    - 5
  - Print (word “total number of blue turtles: ” count foo)
    - Total number of blue turtles: 5



NETLOGO CODING TRICKS

PRACTICE!

- Develop utility functions/reporters if you find yourself using code repeatedly (i.e., create your own primitives)

```
to-report util-random-range [min-extreme max-extreme] ;generates random integer between two values
report random (max-extreme - min-extreme + 1) + min-extreme
end
```

```
to-report util-frequency [an-item a-list] ;counts number of times an item appears in list
   report length (filter [ i -> i = an-item] a-list)
end
```

NETLOGO CODING TRICKS

PRACTICE!

Brute force coding does work and shouldn't be ignored (*but there is probably a primitive or command that will help*)

```
if patch-veg-species = 3 ; Hygrophila
[
  let prob random-float 1
  if prob <= 0.045454545 [set patch-max-darter-density 0]
  if prob > 0.045454545 and prob <= 0.136363636 [set patch-max-darter-density 0.5]
  if prob > 0.136363636 and prob <= 0.181818182 [set patch-max-darter-density 1]
  if prob > 0.181818182 and prob <= 0.25 [set patch-max-darter-density 1.5]
  if prob > 0.25 and prob <= 0.318181818 [set patch-max-darter-density 2]
  if prob > 0.318181818 and prob <= 0.386363636 [set patch-max-darter-density 2.5]
  if prob > 0.386363636 and prob <= 0.454545455 [set patch-max-darter-density 3]
  if prob > 0.454545455 and prob <= 0.590909091 [set patch-max-darter-density 3.5]
  if prob > 0.590909091 and prob <= 0.704545455 [set patch-max-darter-density 5]
  if prob > 0.704545455 and prob <= 0.772727273 [set patch-max-darter-density 5.5]
  if prob > 0.772727273 and prob <= 0.795454545 [set patch-max-darter-density 6.5]
  if prob > 0.795454545 and prob <= 0.840909091 [set patch-max-darter-density 7]
  if prob > 0.840909091 and prob <= 0.863636364 [set patch-max-darter-density 8]
  if prob > 0.863636364 and prob <= 0.886363636 [set patch-max-darter-density 8.5]
  if prob > 0.886363636 and prob <= 0.909090909 [set patch-max-darter-density 12.5]
  if prob > 0.909090909 and prob <= 0.954545455 [set patch-max-darter-density 14.5]
  if prob > 0.954545455 and prob <= 0.977272727 [set patch-max-darter-density 19]
  if prob > 0.977272727 [set patch-max-darter-density 22]
]
```

273 lines of code

```
set Bare-List [0.97877805  5.5352E-05  0.011048256  0.00670866  0.00097973  0.00041514  8.3028E-05  0.000902237]
set Cab-List [0  0.888888889  0.111111111  0  0  0  0  0]
set Hyd-List [0.318079976  0.003341814  0.59886834  0.044848669  0.012000152  0.001291156  7.59503E-05  0.018379979]
set Hyg-List [0.130821962  0.000768787  0.028252931  0.766737139  0.030238965  0.022422961  0.000640656  0.019475943]
set Pot-List [0.194154489  0.005567154  0.078636047  0.265135699  0.412665275  0.011830202  0  0.013917884]
set Sag-List [0.110526316  0  0.010526316  0.470175439  0  0.403508772  0.001754386  0.003508772]
set Val-List [0.632653061  0  0.040816327  0.183673469  0  0  0.142857143  0]
set Ziz-List [0.189089418  0.000820345  0.138638228  0.15709598  0.027891715  0.001230517  0  0.485233798]
```

```
to-report util-partial-sums [#lst]
  set #lst (fput [0] #lst)  ;;prepare for reduce
  ;print word "reporter print:" #lst
  report butfirst reduce [lput (?2 + last ?1) ?1] #lst
end

to-report util-compare-adjacent-pairs-in-list [randnum specieslist]

;print word "species list" specieslist
let post 0
let list1 (butlast specieslist)
let list2 (butfirst specieslist)

ifelse randnum <= first specieslist [set post 0]
  [ifelse randnum > last specieslist [set post position (last specieslist) specieslist]
    [
       (foreach list1 list2 [
          if randnum > ?1 and randnum <= ?2 [set post ((position ? specieslist) + 1)]])
    ]
  ]
report post
end ;util-compare-adjacency-list
```

30 lines of code

TOOLKIT FOR APPLIED MODELING (TAM)

RAPID EQUATION GENERATOR

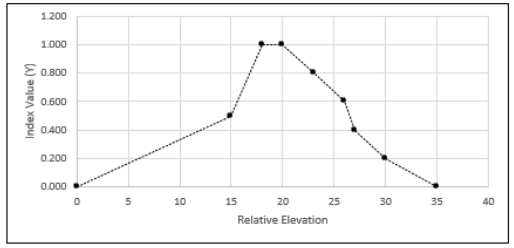
- Can develop equations quickly for model prototyping (linear or logistic curves)
- Focus on the modeling not the math

ENTER DATA INTO HIGHLIGHTED CELLS

Breakpoint #	Relative Elevation	Index Value (Y)
1	0	0.000
2	15	0.500
3	18	1.000
4	20	1.000
5	23	0.8
6	26	0.6
7	27	0.4
8	30	0.2
9	35	0
10		

Values	Intercept	Slope	Equation
0 -15	0.00	0.0333	Y= 0 + (0.0333 * Relative Elevation)
15 -18	-2.00	0.1667	Y= -2 + (0.1667 * Relative Elevation)
18 -20	1.00	0.0000	Y= 1 + (0 * Relative Elevation)
20 -23	2.33	-0.0667	Y= 2.33 + (-0.0667 * Relative Elevation)
23 -26	2.33	-0.0667	Y= 2.33 + (-0.0667 * Relative Elevation)
26 -27	5.80	-0.2000	Y= 5.8 + (-0.2 * Relative Elevation)
27 -30	2.20	-0.0667	Y= 2.2 + (-0.0667 * Relative Elevation)
30 -35	1.40	-0.0400	Y= 1.4 + (-0.04 * Relative Elevation)
35 -			
-			

Comments/References



ENTER DATA INTO HIGHLIGHTED CELLS

Parameters:	Values
MaxX	100
MaxY	0.99
X2	5
Y2	0.01

Breakpoint #	Var A	Var B
1	5	0.01
2	15	0.025888518
3	20	0.041327183
4	25	0.065355017
5	30	0.101868318
6	50	0.439830889
7	70	0.844610262
8	80	0.934644983
9	90	0.974111482
10	100	0.99

Comments/References