

Agent-Based Modeling and Simulation

Collectives

Dr. Alejandro Guerra-Hernández

Universidad Veracruzana

Centro de Investigación en Inteligencia Artificial
Sebastián Camacho No. 5, Xalapa, Ver., México 91000

<mailto:aguerra@uv.mx>
<http://www.uv.mx/personal/aguerra>

August 2019 - January 2020



Credits

- ▶ These slides are completely based on the book of Railsback and Grimm [2], chapter 16.
- ▶ Any difference with this source is my responsibility.



Universidad Veracruzana

Collectives

- ▶ Agent-based models represent a system by modeling its individual agents, but surprisingly many systems include **intermediate levels of organization** between the agents and the system.
- ▶ Agents of many kinds organize themselves into what we call **collectives**: groups that strongly affect both the agents and the overall system. This chapter is about how to model such intermediate levels of organization.



Universidad Veracruzana

Representations

- ▶ As a completely **emergent characteristic** of the agents: the agents have behaviors that allow or encourage them to organize with other agents, the collectives exist only when the agents organize themselves into collectives, and the collectives are given no behaviors or variables of their own.
- ▶ As a separate, **explicit type of entity** in an ABM with its own actions: the modeler specifies how the collectives behave and how agents interact with the collectives.



Implementation

- ▶ NetLogo provides **breeds**: a way to model multiple kinds of turtles or links.
- ▶ Breeds let us do many things, including modelling collectives as explicit entities.



Universidad Veracruzana

Learning Objectives

- ▶ Learn what **collectives** are and several ways of modeling them.
- ▶ Become expert at using NetLogo **breeds** to represent kinds of turtles.
- ▶ Develop experience with **stochastic modeling** by programming and analyzing a new example model, of African wild dogs that live in packs.
- ▶ Become familiar with **logistic functions**, which are particularly useful for representing probabilities and functions that vary nonlinearly between zero and one.



Universidad Veracruzana

Cooperation

- ▶ **Cooperation** among individuals has important advantages, so it is not surprising that systems of all kinds include groups of cooperating individuals.
- ▶ **Example:** In biology, cells of different types form organs and other structures to perform functions that the whole organism depends on.
- ▶ **Example:** In many animal and human societies, family and social groups are very important to the behavior, survival, and reproduction of individuals.
- ▶ **Example:** Economic systems contain many networks and organizations through which individuals and firms cooperate.



Explicit modelling

- ▶ A collective agent typically has its own state variables (especially, a **list of the individuals** in the collective) and traits (often including rules for how individuals are added and removed from the collective).
- ▶ The individuals often have a state variable for **which collective** they belong to and traits for how the collective affects them.
- ▶ This approach of modeling collectives explicitly often adds the least complexity: we can **ignore** all the individual-level detail that, in reality, produces the behavior of the collective.
- ▶ We can think of collectives and their traits as a **simplified** model of their individuals.



Universidad Veracruzana

Representing Collectives via Patches

- ▶ If the individual agents in a model are represented as turtles, sometimes it makes sense to represent collectives of turtles as **patches**.
- ▶ This approach can work, e.g., when a collective is a group of individuals that **occupy the same space**.
- ▶ **Example:** A population of animals or people that live in family or social groups that each occupy a territory (or business, or town).
- ▶ **Patch variables** can describe the collective's characteristics (e.g., the number of members, perhaps broken out by age, sex, etc.), and the primitive **turtles-here** provides an agentset of member turtles.
- ▶ The patch's traits could include **rules for collective behaviors** such as adding and removing individuals.



Universidad Veracruzana

Representing Collectives via Links

- ▶ NetLogo's links might seem **natural** for modeling collectives because we can think of a collective as a group of individuals that are linked in some way.
- ▶ However, NetLogo **does not have** a built-in way to give variables or rules to a network of linked turtles –only to the turtles or links.
- ▶ So links **are not easily used** to represent collectives with their own behaviors.



Using Breeds to Model Multiple Agent Types

- ▶ NetLogo provides breeds for modeling different **kinds** of turtles or links.
- ▶ Breeds are equivalent to **subclasses** in an OOP language: they let you create several **specialized** kinds of turtles that have different characteristics from each other, while still sharing some common variables and traits.



Universidad Veracruzana

Key Points Using Breeds I

- ▶ Breeds must be defined at the **top of the program**.
- ▶ Turtles have a built-in state variable **breed**. If you create a turtle using `create-turtles`, its breed variable is set to “turtles.” Using `create-<breeds>` creates a turtle of the specified breed: `create-wolves` creates a turtle with its value of breed equal to `wolves`.
- ▶ You can **change what breed** a turtle is by simply using `set breed`.
- ▶ Each breed can have its **own unique state variables**, defined via a `<breeds>-own` statement.
- ▶ Each breed still has the **built-in variables** (`xcor`, `ycor`, `color`, etc.) that NetLogo provides for all turtles.
- ▶ The statement `turtles-own` can still be used to define variables that **all breeds** use.



Key Points Using Breeds II

- ▶ If you have defined the breeds “wolves” and “sheep,” you can use `ask wolves [...]` or `ask sheep [...]` to have only wolves or only sheep do something. But `ask turtles [...]` still causes all turtles of all breeds to do something. Likewise, you can use `mean [age] of sheep` and `mean [age] of turtles` to get the average age of just sheep and of all turtles.
- ▶ Several of the `agentset primitives` have `breed versions`. **Example:** `sheep-here` reports an agentset of sheep on a patch, while `turtles-here` reports all turtles on the patch.
- ▶ Breeds do not have their own `contexts`, all are treated as being in `turtle context`. Hence, a wolf can try to run a procedure that you wrote for sheep. However, if a wolf tries to use a variable that was defined only for sheep, an `error` is produced.



Universidad Veracruzana

Similar Breeds

- ▶ Although breeds can be very different, e.g., wolves and sheep, this is not necessarily the case.
- ▶ They can represent slight variations of the same agent to analyse the effect of the variation in the same simulation.
- ▶ **Example:** Growers versus savers in an economy, where the only difference between them is the parameter controlling how rapidly they expand.



Universidad Veracruzana

Individuals as Breeds

- ▶ We can represent individuals as **one breed**, while representing collectives as a **separate breed** with completely different variables and traits.
- ▶ For this, we must **keep track** of which individuals belong to each collective, and which collective each individual belongs to.
- ▶ The trick consists in giving each individual a variable that contains the **collective** he belongs to; and giving each collective a list or an agent set of their **members**.
- ▶ **Updating:** each time an individual leaves or joins a collective:
 - ▶ We need to remove or add the individual from/to the collective in question.
 - ▶ We need to update the individual's variable representing the collective in question.



Universidad Veracruzana

Social Groups

- ▶ Some of the clearest examples of collectives are populations of animals that form **family** or social groups.
- ▶ These animals typically **cooperate** within their collectives to obtain food and other resources.
- ▶ **Example:** **Mating** and **reproduction** can be tightly regulated within the collectives: a group's dominant α male and female mate, but suppress others' reproductive behavior.
- ▶ Yet the behavior of the individuals **strongly affect** their collective: the reproductive output of the α couple depends on
 - ▶ How other members decide whether to **remain** or seek better opportunities elsewhere, and
 - ▶ whether individuals leave or die determines the **size** and **structure** of the group.



Universidad Veracruzana

African Wild Dogs

- ▶ Our example is a model of *Lycaon pictus* which are endangered and carefully managed in a number of nature reserves in South Africa [1].
- ▶ Detailed **observations** of the wild dogs have been collected for many years, so we have extensive empirical information to use:
 - ▶ The **frequency** with which packs produces pups decreases as the total population of dogs increases.
 - ▶ Subordinate adults (non α , two or more years old) often leave their pack as **dispersers** (50% of the time).
 - ▶ When two disperser groups of opposite sex and originating from different packs meet, they usually (64% of the time) **join** to form a new pack.
 - ▶ The annual **mortality** rate varies with dog age and is much higher for dispersing dogs than for those in a pack.



Universidad Veracruzana

Structure Of The Model

- ▶ **Individual dogs** need to be model, since many processes depend on individual variables, e.g., age, sex, and social status.
- ▶ It is clear that no one, but two kinds of **collectives** are important:
 - ▶ Packs
 - ▶ Disperser groups
- ▶ Therefore we need three **breeds**: dogs, packs, and disperser groups.



Stochastic Processes

- ▶ Stochastic traits would be very useful.
- ▶ The behaviors of the dogs and the collectives they form are complex, but **frequencies** and **rates** observed in the field can be used as **probability parameters** for stochastic traits.
- ▶ Reproduction, dispersal decisions, and mortality can all be modeled as **stochastic events** with probabilities based on observed frequencies.

Management Actions

1. To **increase the "carrying capacity"** or the reserve, the maximum number of dogs that are resources to support. This can be achieved by increasing the size of the reserve or by reducing the competition for prey from other predator such as lions.
2. To **reduce the mortality of dispersers**, which is high by, e.g., favoring dispersers encounters.



Universidad Veracruzana

Purpose

- ▶ Evaluating how **persistence** of a wild dog population depends on:
 - ▶ The reserve's **carrying capacity**,
 - ▶ The ability of dispersing dogs to **find** each other, and
 - ▶ The **mortality** risk of dispersing dogs.
- ▶ **Measures** of persistence include the average number of years (over a number of replicate simulations) until the population is extinct, and the percentage of simulations in which the population survives for at least 100 years.



Entities, State Variables, and Scales I

- ▶ Three **kinds of agents**: dogs, packs, and disperser groups.
- ▶ **Dogs** have state variables for age in years, sex, the pack or disperser group they belong to, and social status.
- ▶ **Social status** has one of the following values:

Status	Age	Observations
pup	0	
yearling	1	
subordinate	≥ 2	if it is not an α -dog .
alfa	≥ 2	Chosen randomly, one male and one female.
disperser	≥ 2	Member of a disperser group, instead of a pack.



Entities, State Variables, and Scales II

- ▶ Dog **packs** have no state variable, except for a **list** (or **agentset**) of the dogs belonging to the pack, i.e., members.
- ▶ **Disperser groups** have a state variable for sex (all members are of the same sex) and the agentset of their members.
- ▶ The **time step** is one year.
- ▶ The model is **non spatial**: locations of packs and dogs are not represented.



Process Overview and Scheduling I

1. Age and social status update:

- ▶ The age of all dogs is incremented. Their social status is updated accordingly to their new age.
- ▶ Each pack updates its α male and female: If there is no α of a sex, a subordinate of that sex is randomly selected as alpha.

2. Reproduction:

- ▶ If the pack does not include both an α female and male, no pups are produced.
- ▶ Otherwise, the probability of a pack of producing pups P depends on the total of number dogs in the entire population at the current time step N , modeled as a **logistic function** of N . This function depends on the carrying capacity of the reserve, initially 60 dogs. $P = 0.5$ when N is half of the carrying capacity. $P = 0.1$ when N is the total carrying capacity.



Universidad Veracruzana

Process Overview and Scheduling II

- ▶ If the pack reproduces, the number of pups is drawn from a **Poisson distribution** that has a mean birth rate (pups per pack per year) of 7.9. Sex is assigned to each pup randomly, with a 0.55 probability of being male. Pup age is set to 0.

3. Dispersal:

- ▶ If a pack has no subordinates, then no disperser group is created.
- ▶ If a pack has only one subordinate of its sex, it has a probability of 0.5 of forming a one-member dispersing group.
- ▶ If a pack has 2 or more subordinates of the same sex, they always form a disperser group.
- ▶ Dogs that join a disperser group no longer belong to their original pack, and their social status is set to disperser.



Universidad Veracruzana

Process Overview and Scheduling III

4. Dog mortality is scheduled before pack formation, because mortality of dispersers is high. Whether or not each dog dies is determined stochastically:

Status	Probability
disperser	0.44
yearling	0.25
subordinate	0.20
alfa	0.20
pups	0.12

5. Mortality of collectives, if any pack or dispersal group has no members, it is removed from the model. If any pack contains only pups, the pups die and the pack is removed.



Process Overview and Scheduling IV

6. Pack formation. Disperser groups may meet other disperser groups, and if they meet a group of the opposite sex and from a different pack, they may or may not form a new pack. Each disperser group execute the following steps in **random order**:
 - ▶ Determine how many **times** the group meets another disperser group. This is modeled as a Poisson process with the rate of meeting (mean number of times per year that another group is met) equal to the number of other disperser groups times a **parameter** controlling how often groups meet. This parameter can potentially be greater than 1, but is given a default value of 1.0. The following steps are repeated up to rate of meeting times, stopping if the disperser group selects another to join.
 - ▶ Randomly select other disperser group. It is possible to select the same other group more than once.



Process Overview and Scheduling V

- ▶ If the other group is of the same sex, or originated from the same pack, then do nothing more.
- ▶ If the other group is of the opposite sex and a different pack, then there is a probability of 0.64 that the two groups join into a new pack. If they do not join, nothing else happens.
- ▶ If two disperser groups do join, a **new pack** is created immediately and all the dogs in the two groups become its members. The alpha male and female are chosen randomly; all other members are given a social status of “subordinate.” The two disperser groups are no longer available to merge with remaining groups.



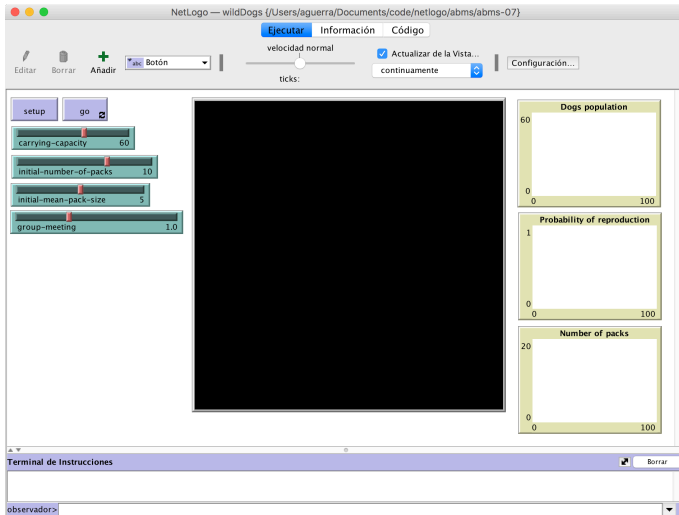
Initialization

- ▶ The model is initialized with 10 packs and no disperser groups.
- ▶ The number of dogs in each initial pack is drawn from a Poisson distribution with mean of 5 (the Poisson distribution is convenient even though its assumptions are not met in this application).
- ▶ The sex of each dog is set randomly with equal probabilities.
- ▶ The age of individuals is drawn from a uniform integer distribution between 0 and 6.
- ▶ Social status is set according to age.
- ▶ The alpha male and female of each pack are selected randomly from among its subordinates; if there are no subordinates of a sex, then the pack has no alpha of that sex.



Universidad Veracruzana

The execution tab



Global variables

- ▶ Two global variables are required:

```
1  globals  
2  [  
3    years-to-simulate ;; set to one hundred by default  
4    prob-reproduction ;; set by a logistic func in reproduce  
5  ]
```



Universidad Veracruzana

Breeds

- Kinds of agents are implemented as breeds, both individuals (dogs) and collectives (packs and disperser-groups):

```
1 | ; Breeds represent collectives in the model
2 |
3 | breed [dogs dog]
4 | breed [packs pack]
5 | breed [disperser-groups disperser-group]
```



Other Variables I

- ▶ You can define variables for the declared dogs breed:

```
1 dogs-own
2 [
3   age
4   sex
5   status
6   my-pack
7   my-disperser-group
8 ]
```

- ▶ As well as packs:

```
1 packs-own
2 [
3   pack-members
4 ]
```

Other Variables II

► And disperser groups:

```
1 disperser-groups-own
2 [
3   group-sex
4   group-members
5 ]
```



Universidad Veracruzana

Setup

```
1  to setup
2    ca
3
4    set years-to-simulate 100 ; number of years to be simulated
5
6    create-packs initial-number-of-packs ; ten by default
7    [
8      initial-setup ; configure a new pack
9    ]
10
11   reset-ticks
12 end
```



Initial setup I

```
1  to initial-setup ; a pack procedure
2      ; set a location and shape just for display
3  setxy random-xcor random-ycor
4  set shape "box"
5
6      ; create a pack of dogs
7  let num-dogs random-poisson initial-mean-pack-size
8
9  hatch-dogs num-dogs
10  [
11      ; first, set display variables
12      set shape "dog"
13      set heading random 360
14      fd 2
15
16      ; now assign dogs variables
```



Initial setup II

```
17  ifelse random-bernoulli 0.5
18    [set sex "male"]
19    [set sex "female"]
20
21  set age random 7
22  update-status
23  set my-pack myself
24  ] ; end of hatch-dogs
25
26  ; initialize the pack's agentset of members
27  set pack-members dogs with [my-pack = myself]
28
29  ; now select the alpha dogs in the pack
30  update-alphas
31  end ;end of initial-setup
```



Random bernoulli does not exist

```
1 | to-report random-bernoulli [ p ]  
2 |   report random-float 1 < p  
3 | end
```



Universidad Veracruzana

Updating the status of a dog

```
1  to update-status ; a dog procedure
2    ifelse age = 0
3      [set status "pup"] ; a pup's age is younger than one year
4      [ifelse age = 1
5        [set status "yearling"] ; a yearling's age is one
6        [if status != "alpha"
7          [set status "subordinate"]
8        ]
9      ] ; older than one
10  end ; set-my-status end
```



Updating alphas I

```
1  to update-alphas ; a pack procedure
2    if not any? pack-members with [sex = "female" and status =
3      "alpha"]
4    [
5      if any? pack-members with [sex = "female" and status =
6        "subordinate"]
7      [
8        ask one-of pack-members with [sex = "female" and status
9          = "subordinate"]
10         [
11           set status "alpha"
12           set size 1.5
13         ]
14      ]
15    ]
```

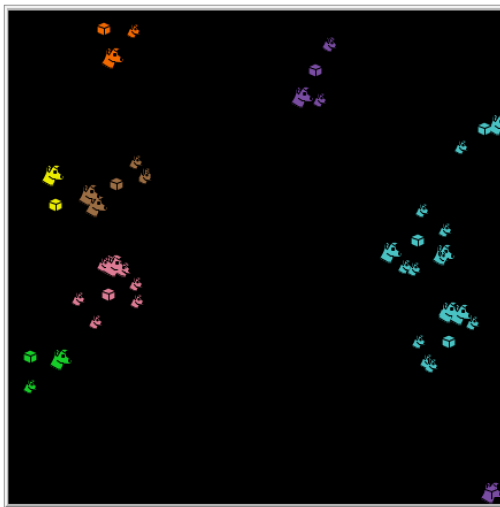


Updating alphas II

```
14  if not any? pack-members with [sex = "male" and status =  
    "alpha"]  
15  [  
16    if any? pack-members with [sex = "male" and status =  
        "subordinate"]  
17    [  
18      ask one-of pack-members with [sex = "male" and status =  
          "subordinate"]  
19      [  
20        set status "alpha"  
21        set size 1.5  
22      ]  
23    ]  
24  ]  
25  end ; end update-alias
```



Testing the set up



Universidad Veracruzana

The Schedule I

```
1  to go
2    tick
3    if ticks >= years-to-simulate [stop]
4
5    set prob-reproduction logFunc count dogs ; a logistic
      function of the current number of dogs
6
7    ; age and social status update
8    ask dogs
9    [
10      set age age + 1 ;; dogs become one year older
11      update-status ;; updated accordingly to their age
12    ]
13
14    ask packs [update-alphas]
15    ask packs [reproduce]
```



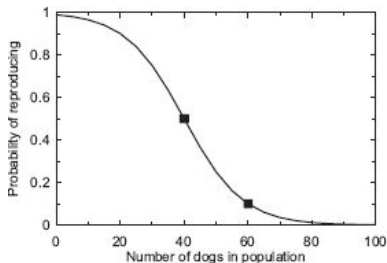
The Schedule II

```
16  ask packs [disperse]
17  ask dogs [dog-die]
18  ask packs [pack-die]
19  ask disperser-groups [if count group-members = 0 [die]]
20  ask disperser-groups [pack-formation]
21  end ; go ends
```



The logistic function

- ▶ They are useful for modeling how the **probability** of some event depends on one or several variables, because:
 - ▶ They produce values in the probability **range**;
 - ▶ They reproduce how probabilities can be very low over a wide range of conditions, very low over a wide range, very high over another range, and **change** sharply between these two ranges.



Universidad Veracruzana

Computation

- ▶ Identify two points in the logistic curve:
 - ▶ $P1 = 30, 0.5$, i.e., when the population of dogs is half the capacity of the reserve, the probability of reproduction is 0.5.
 - ▶ $P2 = 60, 0.1$, i.e., when the population of dogs equals the capacity of the reserve, the probability of reproduction is 0.1.
- ▶ Calculate the following variables:
 - ▶ $D = \ln(P1/(1 - P1))$
 - ▶ $C = \ln(P2/(1 - P2))$
 - ▶ $B = (D - C)/(X1 - X2)$
 - ▶ $A = D - (B \times X1)$
- ▶ Return $P = Z/(1 + Z)$ where $Z = \exp(A + (B \times X))$.



Coding the logistic function

```
1  to-report logFunc [x]
2    let x1 30
3    let p1 0.5
4    let x2 60
5    let p2 0.1
6
7    let d ln (p1 / (1 - p1))
8    let c ln (p2 / (1 - p2))
9    let b (d - c) / (x1 - x2)
10   let a (d - (b * x1))
11
12   let z exp (a + (b * x))
13
14   report z / (1 + z)
15 end ; logFunc ends
```



Testing the function

- ▶ You can test the function using the command center:

```
Command Center
observer> show logFunc 30
observer: 0.5
observer> show logfunc 60
observer: 0.10000000000000002
observer> show logfunc 10
observer: 0.812268223212867
observer> show logfunc 5
observer: 0.8618832502903921
observer>
```



Reproducing I

```
1  to reproduce ; a pack procedure
2    if count pack-members with [status = "alpha"] = 2 ; there
      is a male and female alphas
3    [
4      let number-pups random-poisson 7.9 ;; number of born pups
5
6      if random-bernoulli prob-reproduction
7      [
8        hatch-dogs number-pups
9        [
10         set shape "dog"
11         set age 0 ; new borns are 0 years old
12         set status "pup"
13
14         ifelse random-bernoulli 0.55 ; sex is randomly set
15         [set sex "male"]
```



Reproducing II

```
16         [set sex "female"]
17
18     set my-pack myself
19 ]
20 ]
21 ]
22 end ; reproduce ends
```



Dispersing I

```
1  to disperse ; a pack procedure
2    let male-subordinates pack-members with [status =
      "subordinate" and sex = "male"]
3    let female-subordinates pack-members with [status =
      "subordinate" and sex = "female"]
4
5    if any? female-subordinates
6      [
7        ifelse count female-subordinates = 1 ; creates female
          disperser groups
8          [
9            if random-bernoulli 0.5 [ create-disperser-group-from
              female-subordinates ]
10           ]
11          [
12            create-disperser-group-from female-subordinates
```



Dispersing II

```
13   ]
14 ]
15
16 if any? male-subordinates
17 [
18   ifelse count male-subordinates >= 1 ; creates male
      disperser groups
19   [
20     if random-bernoulli 0.5 [ create-disperser-group-from
      male-subordinates ]
21   ]
22   [
23     create-disperser-group-from male-subordinates
24   ]
25 ]
26 end ; disperse ends
```



Dispersing groups I

```
1  to create-disperser-group-from [some-dogs] ; a pack procedure
2  hatch-disperser-groups 1 ; create one disperser group
3  [
4    set group-members some-dogs ; with some-dogs as members
5    set group-sex [sex] of one-of some-dogs ; all members
      have same sex
6
7    set shape "car"
8    set heading random 360
9    fd 2
10
11   ask some-dogs
12   [
13     set my-disperser-group myself
14     set status "disperser"
15     set color green
```



Dispersing groups II

```
16
17     move-to my-disperser-group
18     set heading [heading] of my-disperser-group
19     fd 1 + random-float 2
20 ] ; end of ask some-dogs
21 ] ; end of hatch
22
23 let dogs-former-pack [my-pack] of one-of some-dogs
24 ask dogs-former-pack
25 [
26     set pack-members pack-members with
27         [status != "disperser"]
28 ]
29 end
```



Dogs' death

```
1  to dog-die
2    ifelse status = "disperser"
3    [ if random-bernoulli 0.44 [die] ]
4    [ifelse status = "yearling"
5      [ if random-bernoulli 0.25 [die] ]
6      [
7        ifelse status = "subordinate" or status = "alpha"
8        [ if random-bernoulli 0.20 [die] ]
9        [
10         if random-bernoulli 0.12 [die] ; pups probability of
            dying
11       ]
12     ]
13   ]
14  end
```



Packs' death

```
1  to pack-die ; a pack procedure
2    ifelse count pack-members = 0
3      [die] ; the pack dies if it has no members
4      [if all? pack-members [status = "pup"]
5        [die] ; the pack dies if all its members are pups
6      ]
7  end
```



Packs' formation I

```
1  to pack-formation ; a disperser-group procedure
2    let num-groups-met random-poisson count disperser-groups *
      group-meeting
3    let our-sex group-sex
4    let our-pack [my-pack] of one-of group-members
5    let our-members group-members
6
7    let new-pack-formed false ; true if a new pack is formed
8
9    repeat num-groups-met
10   [
11     ask one-of disperser-groups
12     [
13       let their-members group-members
14       let their-sex group-sex
15       let their-pack [my-pack] of one-of group-members
```



Universidad Veracruzana

Packs' formation II

```
16
17 if their-sex != our-sex and their-pack != our-pack
18 [
19   if random-bernoulli 0.64
20   [
21     set new-pack-formed true
22     hatch-packs 1
23     [
24       set shape "box"
25       setxy random-xcor random-ycor
26       set pack-members (turtle-set their-members
27                           our-members)
28       ask pack-members
29       [
30         set shape "dog"
31         set heading random 360
32         fd 2
```



Packs' formation III

```
32         set status "subordinate"
33         set my-pack myself
34     ]
35     set pack-members dogs with [my-pack = myself]
36     update-alphas
37     ; show-pack ; to debug
38 ]
39 die ; the other disperser-group dies
40 ]
41 ]
42 ]
43 ]
44 if new-pack-formed [die] ; the former disperser-group dies
45 end
```

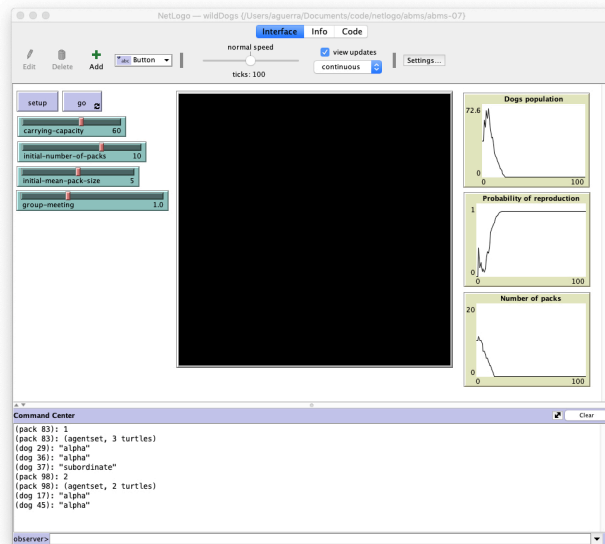


Debugging

```
1  to show-pack ; a debug procedure for pack-formation
2    show ticks
3    show pack-members
4    ask pack-members
5    [
6      show status
7    ]
8  end ; show-pack ends
```



Output



Universidad Veracruzana

Representations I

- ▶ Collectives can be modeled as an **emergent** trait of the individuals: individuals are modeled in such a way that they can form collectives; and the collectives a strictly an outcome of the model. **Example.** Flocking and the butterflies.
- ▶ When collectives have important and even moderately complex behavior, it is typically necessary to model them **explicitly** as a separate kind of entity with its own state variables and traits. In NetLogo, one way to model collectives explicitly is to assign the properties and traits of collectives to **patches**. This approach can work well when the collectives each occupy a piece of some landscape, and in models that are nonspatial so patches can represent only the collectives.



Universidad Veracruzana

Representations II

- ▶ In other models, though, the **breeds** feature of NetLogo is especially useful for representing collectives because it lets us create multiple kinds of turtle (or link).

Cost

- ▶ Collectives are often essential in ABMs, but their use has a cost in **addition to programming**.
- ▶ Because collectives are an **additional level of organization**, we now need to model and analyze how the properties and behavior of collectives and individuals affect each other, as well as how the system affects both collectives and individuals.



Universidad Veracruzana

Referencias I



M Gusset et al. "Dogs on the catwalk: Modelling re-introduction and traslocation of endangered wild dogs in South Africa". In: *Biological Conservation* 142.2009 (2009), pp. 2774–2781.



SF Railsback and V Grimm. *Agent-Based and Individual-Based Modeling*. Princeton, New Jersey, USA: Princeton University Press, 2012.



Universidad Veracruzana