

Agent-Based Modeling and Simulation

Collectives

Dr. Alejandro Guerra-Hernández


Instituto de Investigaciones en Inteligencia Artificial
Universidad Veracruzana
*Campus Sur, Calle Paseo Lote II, Sección Segunda No 112,
Nuevo Xalapa, Xalapa, Ver., México 91097*

`mailto:aguerra@uv.mx`
`https://www.uv.mx/personal/aguerra/abms`

Doctorado en Inteligencia Artificial 2024



Credits

- ▶ These slides are based on the book of Railsback and Grimm [4], chapter 16.
- ▶ Any difference with this source is my responsibility.
- ▶ This work is licensed under [CC-BY-NC-SA 4.0](https://creativecommons.org/licenses/by-nc-sa/4.0/) 
- ▶ To view a copy of this license, visit:

<https://creativecommons.org/licenses/by-nc-sa/4.0/>



Collectives

- ▶ ABMs 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 **collectives**, *i.e.*, groups or aggregations of agents that strongly affect both the agents and the overall system.
- ▶ Our course of **SMA** approaches the foundations of these collectives in terms of **interaction situations** [2] and **social actions** [1]. The former is quite relevant here.

Two possible representations

- Emergent.** The **behavior of the agents** allow or encourage them to organize with other agents. The collectives exist only when the agents organize themselves, so that they are given no behaviors or variables of their own.
- Explicit.** The modeler specifies how the collectives, as **entities** in the ABM, 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**.



Learning Objectives

- ▶ Learn what **collectives** are and several ways of modeling them.
- ▶ Use NetLogo **breeds** to represent kinds of turtles.
- ▶ Use NetLogo **links** to represent relations among turtles of different breeds, which is one way to program collectives.
- ▶ Use **stochastic modeling** by programming and analyzing a new example model of African wild dogs that live in packs.
- ▶ Use **logistic functions** for representing probabilities and functions that vary nonlinearly between zero and one.

Cooperation

- ▶ **Cooperation** among individuals has important advantages, so 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 networks and organizations through which individuals and firms cooperate.



Explicit modelling

- ▶ A **collective agent** typically includes:
 - State variables:** especially, a list of the **members** of the collective; and
 - Sub-models:** often including rules for how individuals are **added** and **removed** from the collective.
- ▶ The **individual agents** often have:
 - State variable:** especially the **collectives** they belong to; and
 - Sub-models:** for how the collective affects them.
- ▶ This approach often adds the **least complexity**: we can **ignore** all the individual-level detail that produces the behavior of the collective.

Representing Collectives via Patches

- ▶ When the collective is a group of turtles that occupy the **same space**, it makes sense to represent collectives as **patches**
- ▶ **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.
- ▶ The primitive **turtles-here** provides an agentset of member turtles.
- ▶ The patch's sub-models could include rules for **collective behaviors** e.g., adding and removing individuals.

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.
- ▶ 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. We discuss only **turtle breeds**.
- ▶ 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 sub-models.

Key Points of 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.
- ▶ **Example.** `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.



Key Points of Using Breeds II

- ▶ 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.
- ▶ If you have defined the breeds `wolves` and `sheep`, you can use `ask wolves` (or `sheeps`) to have only wolves or only sheep do something. But `ask turtles` still causes all turtles of all breeds to do something.
- ▶ **Example.** 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.



Key Points of Using Breeds III

- ▶ 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**.
- ▶ **Example.** 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.



Similar Breeds

- ▶ Although breeds can be **very different**, e.g., wolves and sheep, this is not necessarily the case.
- ▶ They can also 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.

Individuals as Breeds

- ▶ Individuals and collectives can be represented as **separate** breeds, with completely different variables and sub-models.
- ▶ For this, we must keep track **membership**:
 - ▶ Each individual has a variable that denotes its collective; and
 - ▶ Each collective has a variable that denote their members.
- ▶ Each time an individual leaves or joins a collective **updating** is as follows:
 - ▶ Individuals are removed/added from/to the collective in question;
 - ▶ The individual's variable representing the collective in question must be updated.

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.
- ▶ However, some behaviors can be tightly regulated within the collectives, e.g., a group's dominant α male and female **mate**, but suppress others' reproductive behavior.
- ▶ Yet the **behavior of the individuals** strongly affect their collective, e.g., 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.



African Wild Dogs

- ▶ *Lycaon pictus* are endangered and carefully managed in nature reserves in South Africa [3]. Detailed **observations** of the wild dogs offer 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** (the only non- α 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.



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 sub-models 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 sub-models.
- ▶ 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** and therefore the relation between the total number of dogs and the probability of each pack reproducing each year.
2. To **increase the size of the reserve** could reduce the probability that disperser groups encounter each other; and
3. Other management actions could reduce the **probability that dispersing dogs die** before forming a new pack.

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.

Usefulness

- ▶ Usefulness of the original model for this purpose is evaluated using five **patterns** observed in the real wild dog population by Gusset et al. [3].
- ▶ The patterns **relevant to this version** of the model include:
 - ▶ The range (mean and standard error among years) in **pack size**,
 - ▶ **Sex** ratio,
 - ▶ Proportion of the population that are **pups**, and
 - ▶ Proportion of disperser groups that fail to form **new packs**.

Entities, State Variables, and Scales I

- ▶ Kinds of **agents** include dogs, packs, and disperser groups.
- ▶ **Dogs** have state variables for:
 - ▶ Age in years,
 - ▶ Sex,
 - ▶ Social status, set as follows:

Status	Age	Observations
pup	0	
yearling	1	
subordinate	≥ 2	if it is not an α -dog .
alpha	≥ 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 variables.
- ▶ **Disperser groups** have a state variable for sex (all members are of the same sex) and an identifier of the natal pack of their members.
- ▶ Each **time step** represents one year.
- ▶ The model is **non spatial**, locations of packs and dogs are not significant and are only used as visual clues.

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 (Table, slide 24).
- ▶ Each pack updates its α male and female: If there is no α of a sex, a subordinate of that sex (if there is one) is randomly chosen.

2. Reproduction:

- ▶ If the pack does not include both an α female and male, no pups are produced, otherwise
- ▶ The probability P of a pack of producing pups depends on the total population at the current time step N , modeled as a **logistic function** of N , depending 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.



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 two 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.



Process Overview and Scheduling III

4. Dog mortality is scheduled before pack formation, because mortality of dispersers is high:

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 that meet other disperser groups of the opposite sex and different original pack, might form a new pack. Each disperser group execute the following steps in **random order**:
 - ▶ Determine how many times the group meets another disperser group, 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** (≥ 1.0) controlling how often groups meet. 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.



The execution tab

NetLogo — wildDogs (/Users/aguerra/Documents/code/netlogo/abms/abms-07)

Ejecutar Información Código

velocidad normal Actualizar de la Vista...
ticks: continuamente Configuración...

Editar Borrar Añadir Botón

setup go

carrying-capacity 60

initial-number-of-packs 10

initial-mean-pack-size 5

group-meeting 1.0

Dogs population

Probability of reproduction

Number of packs

Terminal de Instrucciones Borrar

observador>



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 the go procedure  
5 | ]
```



Breeds

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

```
1 | ; Breeds represent collectives and individuals 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 | ]
```

- ▶ As well as **disperser groups**:

```
1 | disperser-groups-own
2 | [
3 |   sex
4 |   natal-pack-ID
5 | ]
```



Membership to packs and disperser groups

- ▶ How does a dog know which pack or disperser group it **belongs** to, and how do packs and groups know which dogs are in them?
- ▶ By creating an **undirected link** between each dog and its pack (or disperser group).
- ▶ Packs and disperser groups can refer to their **member dogs** using the primitive **link-neighbors**.
- ▶ Links are **automatically removed** whenever the turtle at either end of them dies.
- ▶ So a link is **added** whenever a dog is created or joins a pack or disperser group, a dog can always identify its collective and the collective can always identify its members.

Observations

- ▶ There is one association that links **are not useful for**: because two disperser groups cannot combine into a new pack if they both originated from the same pack, disperser groups must somehow remember which **pack they came from**.
- ▶ We could try giving each disperser group a link to the pack that produced it, but if that pack died before the disperser group did, then the link would disappear. Instead, we use the disperser group variable **natal-pack-ID** to hold the **who** number of the pack where its members were born.

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 ; set at executing tab, ten by
   default
7   [
8     pack-setup ; configure a new pack
9   ]
10
11  reset-ticks
12 end
```



Pack configuration I

```
1  to pack-setup ; a pack procedure
2    ; set a location and shape just for display
3    setxy random-xcor random-ycor
4    set shape "house"
5    set color one-of remove red base-colors
6
7    ; create a pack of dogs
8    let num-dogs random-poisson initial-mean-pack-size
9
10   hatch-dogs num-dogs
11   [
12     ; first, set display variables
13     set shape "dog"
14
15     set heading random 360
16     fd 2
17
18     ; now assign dogs variables
19     ifelse random-bernoulli 0.5
20     [set sex "male"]
```

Pack configuration II

```
21     [set sex "female"]
22
23     set age random 7
24     set-my-status
25
26     ; create link between the dog and its pack
27     create-link-with myself
28 ] ; end of hatch-dogs
29
30 ; now select the alpha dogs in the pack
31 update-pack-alphas
32 end ;end of setup-packs
```



Random bernoulli implementation

- ▶ The random-bernoulli function can be defined as:

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

Updating the status of a dog

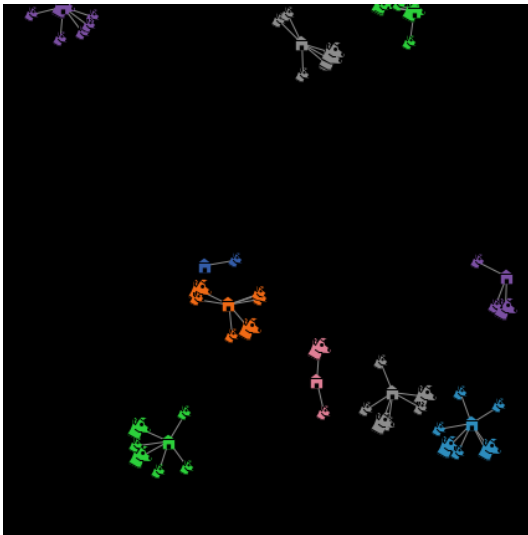
```
1  to set-my-status ; a dog procedure
2    (ifelse
3      age = 0
4      [
5        set status "pup"
6      ] ; a pup's age is younger than one year
7      age = 1
8      [
9        set status "yearling"
10     ] ; a yearling's age is 1
11     ; elsecommands
12     [if status != "alpha"
13       [set status "subordinate"]]
14   ]
15 )
16 end ; set-my-status end
```



Updating alphas I

```
1  to update-pack-alphas ; a pack procedure
2  let pack-members link-neighbors
3  if not any? pack-members with [sex = "female" and status = "alpha"]
4  [
5    let female-subordinates pack-members with [sex = "female" and status =
6      "subordinate"]
7    if any? female-subordinates
8    [
9      ask one-of female-subordinates
10     [
11       set status "alpha"
12       set size 1.5
13     ]
14   ]
15
16   ; The same for males
17   ; ...
18 end ; end update-alphas
```

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 log-func 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    set-my-status ;; updated accordingly to their age
12  ]
13
14  ask packs [update-pack-alphas]
15  ask packs [reproduce]
16  ask packs [disperse]
17  ask dogs [do-mortality]
18  ask packs [do-pack-mortality]
19  ask disperser-groups [if not any? link-neighbors [die]]
```

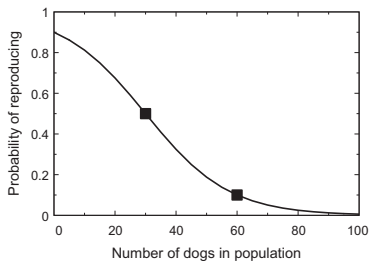
The Schedule II

```
20 ask disperser-groups [do-pack-formation]
21
22 ; Now some defensive programming
23 ; All dogs should be linked to a pack or disperser group,
24 ; and all collectives should be linked to dogs.
25
26 ask turtles with [not any? link-neighbors]
27 [
28   error (word "There is a " breed
29     " with no links; who number: " who)
30 ]
31 ask turtles with [any? link-neighbors with [breed = [breed] of myself]]
32 [
33   error (word breed " " who " is linked to another " breed)
34 ]
35 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.



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   let pack-members link-neighbors
3   if count pack-members with [status = "alpha"] = 2 ; there is a male and
      female alphas
4   [
5     let number-pups random-poisson 7.9 ;; number of born pups
6
7     if random-bernoulli prob-reproduction
8     [
9       hatch-dogs number-pups
10      [
11        set shape "dog"
12
13        set age 0 ; new borns are 0 years old
14        set status "pup"
15
16        ifelse random-bernoulli 0.55 ; sex is randomly set
17        [set sex "male"]
18        [set sex "female"]
19
```

Reproducing II

```
20         create-link-with myself
21     ]
22 ]
23 ]
24 end ; reproduce ends
```


Dispersing

```
1  to disperse ; a pack procedure
2    ; First, identify the subordinates and stop if none
3    let my-subordinates link-neighbors with [status = "subordinate"]
4    if not any? my-subordinates [stop]
5    ; Now check females
6    if count my-subordinates with [sex = "female"] = 1
7      [
8        if random-bernoulli 0.5
9          [
10           create-disperser-group-from my-subordinates with [sex = "female"]
11         ]
12       ]
13    if count my-subordinates with [sex = "female"] > 1
14      [
15        create-disperser-group-from my-subordinates with [sex = "female"]
16      ]
17    ; same for male
18  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 sex [sex] of one-of some-dogs ; all members have same sex
5   set natal-pack-ID [who] of myself
6
7   set shape "car"
8   set color [color] of myself
9   setxy random-xcor random-ycor
10  set heading random 360
11  fd 2
12
13  ask some-dogs
14  [
15    ask my-links [die]
16    create-link-with myself
17    set status "disperser"
18
19    move-to myself
20    set heading [heading] of myself
```

Dispersing groups II

```
21     fd 1 + random-float 2
22   ] ; end of ask some-dogs
23 ] ; end of hatch
24
25 end ; to create-disperser-group-from
```

Dogs' death

```
1  to do-mortality
2  (ifelse
3    status = "disperser"
4    [
5      if random-bernoulli 0.44 [die]
6    ]
7    status = "yearling"
8    [
9      if random-bernoulli 0.25 [die]
10   ]
11   status = "subordinate" or status = "alpha"
12   [
13     if random-bernoulli 0.20 [die]
14   ]
15   ; else commands
16   [
17     if random-bernoulli 0.12 [die] ; pups probability of dying
18   ]
19 )
20 end
```



Packs' death

```
1  to do-pack-mortality ; a pack procedure
2  let pack-members link-neighbors
3  ifelse any? pack-members
4  [if all? pack-members [status = "pup"]
5  [
6  ask pack-members [die]
7  die ; the pack dies if all its members are pups
8  ]
9  ]
10 ; the pack dies if it has no members
11 [die]
12 end
```



Packs' formation I

```

1  to do-pack-formation ; a disperser-group procedure
2  let num-groups-met random-poisson count disperser-groups * group-meeting
3  let new-pack-formed false ; true if a new pack is formed
4  let our-members link-neighbors
5  let our-sex sex
6  let our-natal-pack-ID natal-pack-ID
7
8  repeat num-groups-met
9  [
10   ask one-of disperser-groups
11   [
12    let their-members link-neighbors
13
14    if sex != our-sex and natal-pack-ID != our-natal-pack-ID
15    [
16     if random-bernoulli 0.64
17     [
18      set new-pack-formed true
19      hatch-packs 1
20     [

```

Packs' formation II

```
21     set shape "house"
22     set color red
23     setxy random-xcor random-ycor
24     let pack-members (turtle-set their-members our-members)
25     ask pack-members
26     [
27       move-to myself
28       set heading random 360
29       fd 2
30       set status "subordinate"
31       ask my-links [die]
32       create-link-with myself
33     ]
34     update-pack-alphas
35     ; show-pack ; to debug
36   ]
37   die ; the other disperser-group dies
38 ]
39 ]
40 ]
41 ]
```



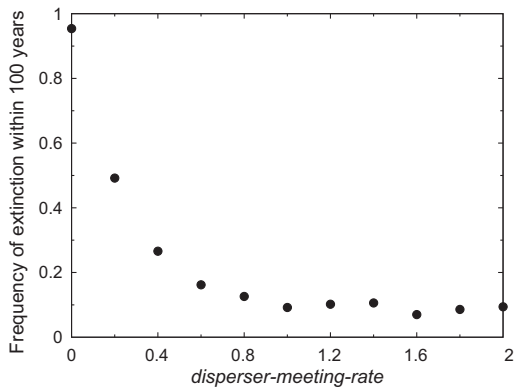
Packs' formation III

```
42 |   if new-pack-formed [die] ; the former disperser-group dies
43 |   end ; do-pack-formation
```


Dispersers meeting rate effect

- ▶ Some reserves have fences to keep dogs from leaving; adding or removing fences could increase or decrease the **meeting rate** of disperser groups.
- ▶ How important is this meeting rate to the **persistence** of the total dog population?
- ▶ **Exercise.** Do a **sensitivity experiment** that varies the value of disperser-meeting-rate over a wide range. How frequently the dog population goes extinct in hundred-year simulations: how many times, out of 500 replicate simulations, there are no dogs alive after 100 years.

Expected plot



Observations

- ▶ **Increasing** the meeting rate above the baseline value (1.0) would not benefit population persistence immensely.
- ▶ However, **reducing** the ability of dispersers to meet is predicted by the model to have stronger negative effects.
- ▶ This analysis could indicate to managers that they should avoid actions that make it harder for disperser groups to meet, while not assuming that improving disperser meeting would make the population substantially less likely to go extinct.



Representations I

- ▶ One way to model collectives is as an **emergent behavior** of the individuals: individuals are modeled in such a way that they can form collectives, and the collectives are strictly an outcome of the model. **Example.** The butterflies.
- ▶ When collectives have important and even moderately complex behavior of themselves, 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 sub-models of collectives to **patches**.

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 turtles (or links).
- ▶ Further, NetLogo's **links** are a simple and relatively reliable way to represent the relations among individuals and their collectives.

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.

Referencias I

- [1] C Castelfranchi. “Modelling Social Action for AI Agents”. In: *Artificial Intelligence* 103.1998 (1998), pp. 157–182.
- [2] J Ferber. *Les Systèmes Multi-Agents: vers une intelligence collective*. Paris, France: InterEditions, 1995.
- [3] 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.
- [4] SF Railsback and V Grimm. *Agent-Based and Individual-Based Modeling*. Second. Princeton, NJ, USA: Princeton University Press, 2019.