Math Decisions 2019/2020
NetLogo: the *Ethnocentrism* model, piece by piece

Alice Raffaele, Romeo Rizzi

Università degli Studi di Verona

April 22, 2020

"Anything you can do, we can do it better"

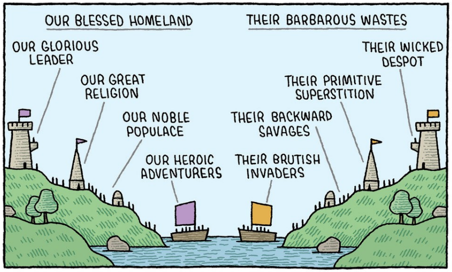
Ethnocentrism is the universal tendency for any culture to see its own values and practices as natural and correct.

Mmm. perfect! Really. no cuisine is as good as ours.
Buzzle.com
Her belief that her cuisine is the best is a classic example of ethnocentrism.

Until it's just a matter of cuisine and best dishes, ok, but...

Definition of ethnocentrism

...but unfortunately it is not:



Definition of W.G. Sumner (1906): " *The view of things in which one's group is the center of everything, and others are scaled and rated with reference to it. Each group nourishes its own pride and vanity, boasts itself superior, exalts its own divinities and looks with contempt on outsiders.*"

Hammond and Axelrod (2006), *The Evolution of Ethocentrism*

An attitude that includes to see:

- an **in-group**: virtuous and superior, one's own standards of value as universal;
- **out-groups**: contemptible and inferior.

Behaviors associated with ethnocentrism:

- **cooperative relations within the group**, based on *group boundaries* typically defined by one or more observable characteristics (e.g., language, accent, physical features, or religion) and strong territorial component;
- **absence of cooperative relations with out-groups**; ethnic conflicts and hostility (xenophobia).

They focus on ethnocentric behavior as **in-group favoritism**.

What Hammond and Axelrod want to verify

- Extensive empirical evidence from psychology suggests the prevalence of a strong individual **predisposition toward bias in favor of in-groups**, observed even with minimal cognition and very abstract social input;
- Behaviors favoring in-groups are also found to be widespread even when they are individually costly and in absence of opportunities for reciprocity or direct self-interested gain.

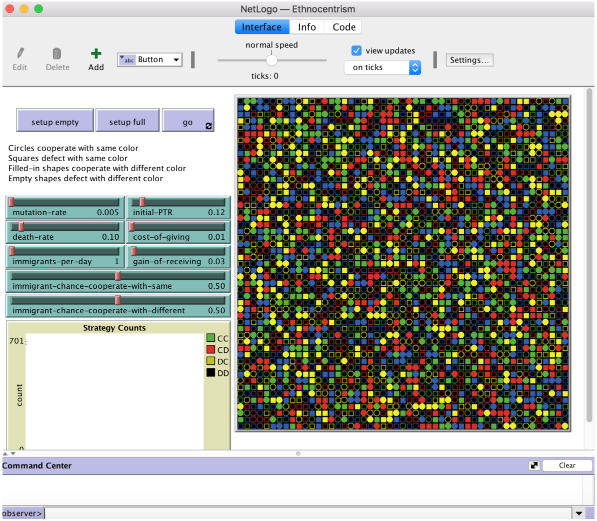
Hammond and Axelrod's results

- Ethnocentric behavior can emerge from their model of local competition between individuals;
- The dominance of ethnocentrism is surprisingly robust to a wide range of changes in the parameters and structure of the model;
- Ethnocentrism can support contingent cooperation in the form of in-group favoritism without requiring mechanisms such as reciprocity, reputation, conformity or leadership.

They use an **evolutionary model** focused on the fundamental dynamics that in-group favoritism can be useful, exploiting an **agent-based simulation technique**.

Ethnocentrism with NetLogo

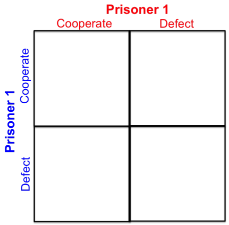
Hammond and Axelrod's model is actually already present in NetLogo Models Library, but we are going to rebuild it, piece by piece.



Circles cooperate with same color
Squares defect with same color
Filled-in shapes cooperate with different color
Empty shapes defect with different color

mutation-rate 0.005
initial-PTR 0.12
death-rate 0.10
cost-of-giving 0.01
immigrants-per-day 1
gain-of-receiving 0.03
immigrant-chance-cooperate-with-same 0.50
immigrant-chance-cooperate-with-different 0.50

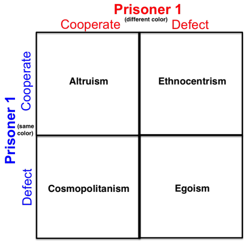
Prisoner's dilemma framework



- Agents compete for limited space via *Prisoner Dilemma*'s type interactions:
  - Cooperation is individually costly;
  - Any opportunity for direct reciprocity is removed, using *one-move prisoner's dilemma* variant.
- "Ethnocentric" agents treat agents within their group more beneficially than those outside their group;
- The model includes a mechanism for inheritance (genetic or cultural) of strategies.

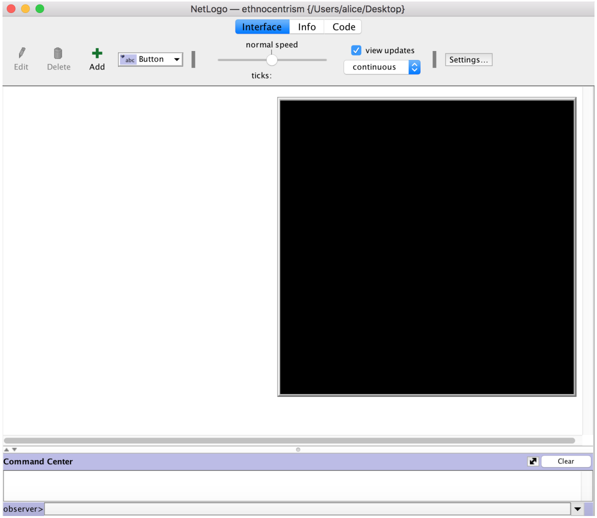
Model settings

- Interaction and propagation of strategies strictly local in 2D space;
- Three traits for each agent:
  1. *tag* - a label that specifies the group membership of an agent, chosen among four predefined colors, and is the only trait observable;
  2. *cooperate/defect (same color)* - what the agent does when it meets someone with its own color;
  3. *cooperate/defect (different color)* - what the agent does when it meets someone with another color.
- The agent **strategy** is not linked to the tag but just defined by second and third traits → 4 strategies:



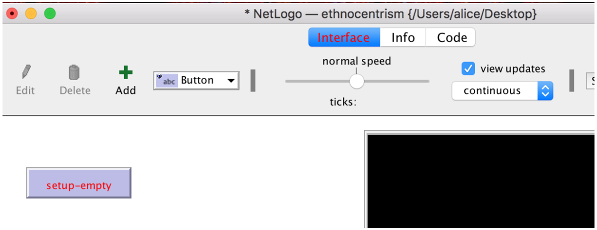
A new model

Let's start creating a new model in NetLogo:

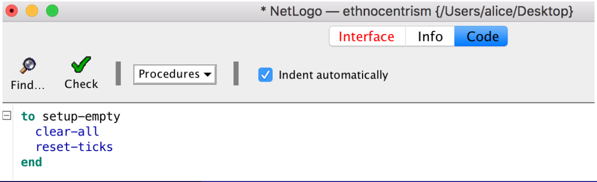


Setup (empty)

We add a **setup-empty** button and its related procedure in the Code tab:



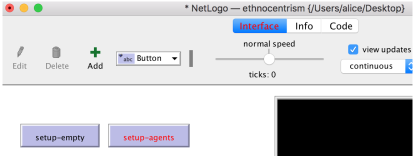
We just want to create an empty grid (*clear-all*) and then reset to zero the ticks counter (*reset-ticks*).



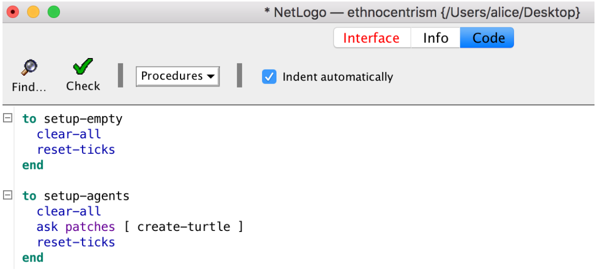
```
to setup-empty
  clear-all
  reset-ticks
end
```

Setup-agents

But we want to populate our grid, thus we add a **setup-agents** button:



We create agents representing people through the command **ask patches [create-turtle]**:



```
to setup-empty
  clear-all
  reset-ticks
end

to setup-agents
  clear-all
  ask patches [ create-turtle ]
  reset-ticks
end
```

Create-turtles

We need to characterize our agents as we said few slides before:

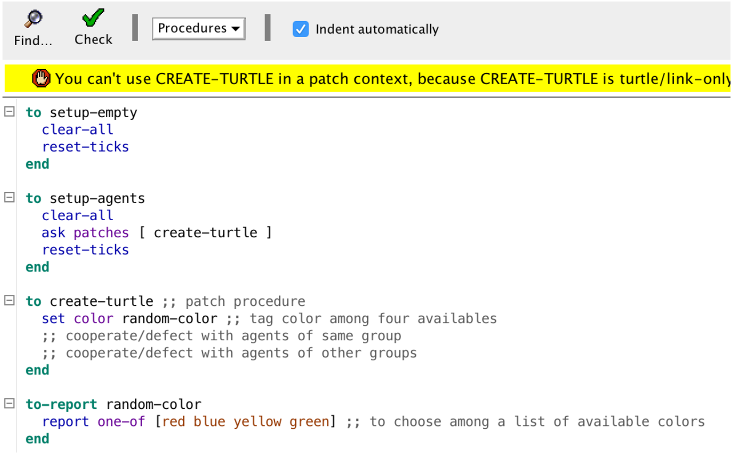
```
to setup-agents
  clear-all
  ask patches [ create-turtle ]
  reset-ticks
end

to create-turtle ;; patch procedure
  ;; tag color among four availables
  ;; cooperate/defect with agents of same group
  ;; cooperate/defect with agents of other groups
end
```

We assign a color to each turtle randomly:

```
to create-turtle ;; patch procedure
  set color random-color ;; tag color among four availables
  ;; cooperate/defect with agents of same group
  ;; cooperate/defect with agents of other groups
end

to-report random-color
  report one-of [red blue yellow green] ;; to choose among a list of available colors
end
```

Create-turtles - Error

If we check the code, it rises the following error:



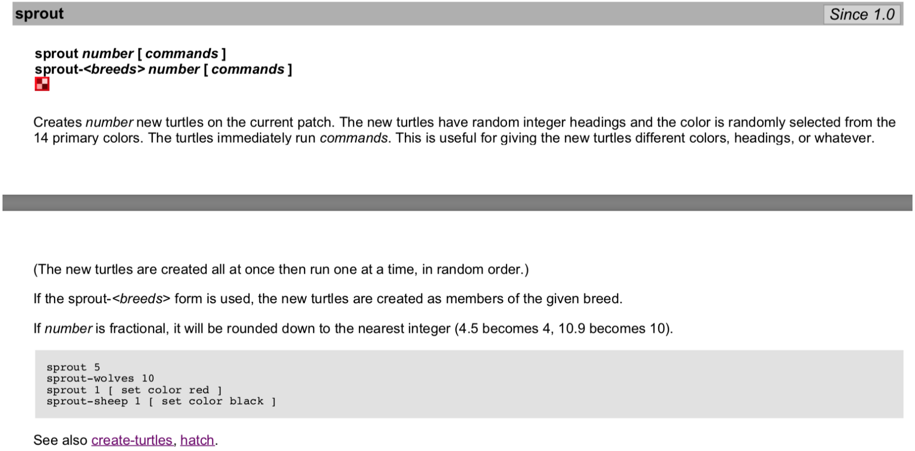
You can't use CREATE-TURTLE in a patch context, because CREATE-TURTLE is turtle/link-only

```
to setup-empty
  clear-all
  reset-ticks
end

to setup-agents
  clear-all
  ask patches [ create-turtle ]
  reset-ticks
end

to create-turtle ;; patch procedure
  set color random-color ;; tag color among four availables
  ;; cooperate/defect with agents of same group
  ;; cooperate/defect with agents of other groups
end

to-report random-color
  report one-of [red blue yellow green] ;; to choose among a list of available colors
end
```

The problem is given by the fact that we should assign a random color to **each** turtle

Sprout

We can use the **sprout** command, described in the NetLogo guide as follows:



sprout *Since 1.0*

sprout *number* [*commands*]
sprout-*<breeds>* *number* [*commands*]

Creates *number* new turtles on the current patch. The new turtles have random integer headings and the color is randomly selected from the 14 primary colors. The turtles immediately run *commands*. This is useful for giving the new turtles different colors, headings, or whatever.

(The new turtles are created all at once then run one at a time, in random order.)

If the sprout-<*breeds*> form is used, the new turtles are created as members of the given breed.

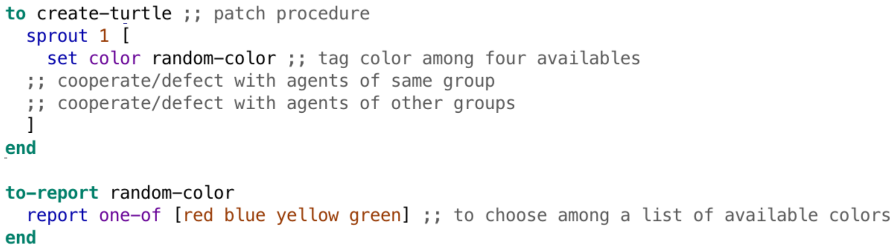
If *number* is fractional, it will be rounded down to the nearest integer (4.5 becomes 4, 10.9 becomes 10).

```
sprout 5
sprout-wolves 10
sprout 1 [ set color red ]
sprout-sheep 1 [ set color black ]
```

See also create-turtles, hatch.

Create-turtles with sprout

Here it is the create-turtles procedure fixed:



```
to create-turtle ;; patch procedure
  sprout 1 [
    set color random-color ;; tag color among four availables
  ;; cooperate/defect with agents of same group
  ;; cooperate/defect with agents of other groups
  ]
end

to-report random-color
  report one-of [red blue yellow green] ;; to choose among a list of available colors
end
```

Create-turtles - Cooperation/defection with other agents

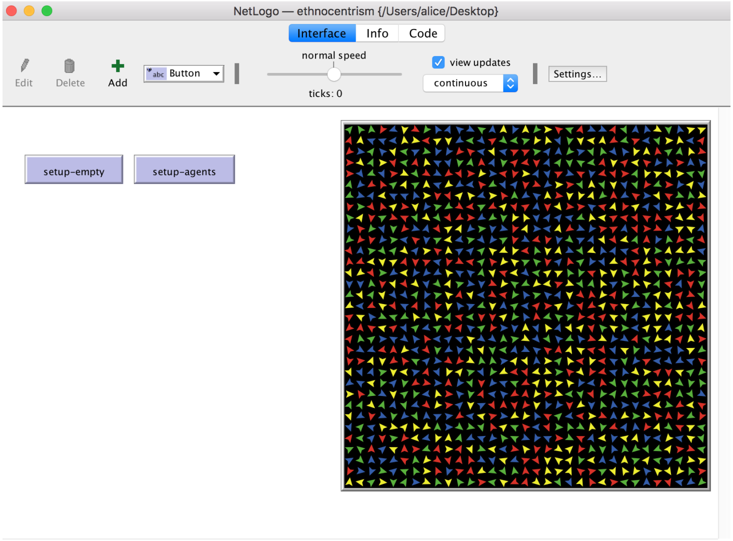
We continue with agents' features, randomly selecting two values in $(0, 1)$ to establish if it cooperates or not with agents of its group and others:

```
turtles-own [ cooperate-same? cooperate-diff? ]

to setup-empty
  clear-all
  reset-ticks
end

to setup-agents
  clear-all
  ask patches [ create-turtle ]
  reset-ticks
end

to create-turtle ;; patch procedure
  sprout 1 [
    set color random-color ;; tag color among four availables
    set cooperate-same? random-float 1.0 ;; cooperate/defect with agents of same group
    set cooperate-diff? random-float 1.0 ;; cooperate/defect with agents of other groups
  ]
end

to-report random-color
  report one-of [red blue yellow green] ;; to choose among a list of available colors
end
```

In order to do so, we introduce two turtles variables: **cooperate-same?** and **cooperate-diff?** (we didn't define variable for the color because it is predefined).

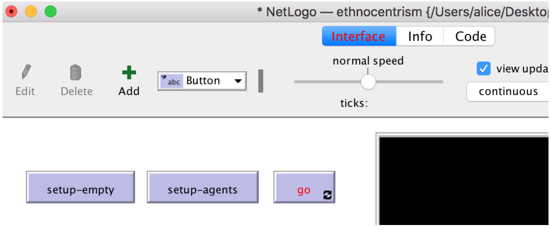
Our model - Version 0.1

And this is how our model looks like after these initial steps:



Go

We add a **go** button:



And its related procedure in the Code tab (at least, its skeleton):

```
to go ;; the main routine
  ;; make turtles interact
  ;; make turtles reproduce
  ;; some turtles may die
end
```

```
to go ;; the main routine
  ask turtles [ interact ];; make turtles interact
  ask turtles [ reproduce ];; make turtles reproduce
  ;; some turtles may die
  tick
end
```

Interact

Agents meet other people: let's call these *immigrants*. At each interaction, an agent compares its color with the immigrant's one and, according to its probabilities to cooperate with same group and other groups, we update the related statistics:

```
to interact  ;; turtles procedure
  ask turtles [
    set meet meet + 1 ;; update statistics on number of people met
    if color = [color] of myself [
      set meet-same meet-same + 1 ;; update statistics on number of people met with same color
      if [cooperate-same?] of myself >= 0.5 [
        set coop-same coop-same + 1
        ;; how much does it cost me?
        ;; how much do I gain?
      ]
    ]
    if color != [color] of myself [
      set meet-diff meet-diff + 1 ;; update statistics on on number of people met with different color
      ifelse [cooperate-diff?] of myself >= 0.5 [
        set coop-diff coop-diff + 1
        ;; how much does it cost me?
        ;; how much do I gain?
      ]
      [
        set def-diff def-diff + 1
      ]
    ]
  ]
end
```

Interact - Statistics

To count the number of interactions per tick and also the specific number of meetings with people having the same color or a different one, we introduce the following global variables:

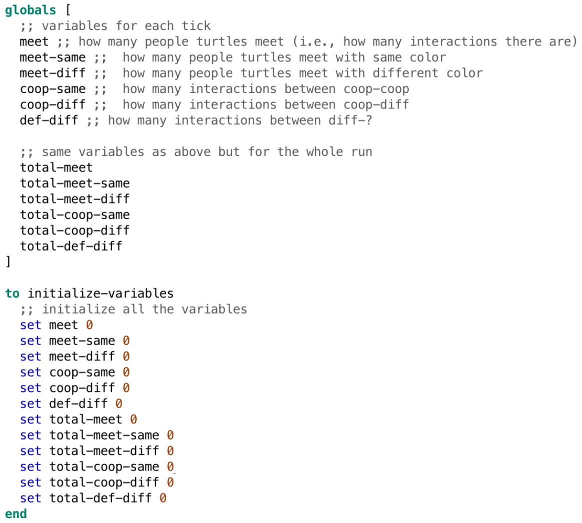
```
globals [
  meet ;; how many people turtles meet (i.e., how many interactions there are)
  meet-same ;;  how many people turtles meet with same color
  meet-diff ;;  how many people turtles meet with different color
  coop-same ;;  how many interactions between coop-coop
  coop-diff ;;  how many interactions between coop-diff
  def-diff ;; how many interactions between diff-?
]
```

We also add similar counters for the total values per run and we introduce a procedure to **reset-stats** each time we press a setup button (we call this in the two setup procedures):

```
to reset-stats
  set meet 0
  set meet-same 0
  set meet-diff 0
  set coop-same 0
  set coop-diff 0
  set def-diff 0
end
```

Initialize variables

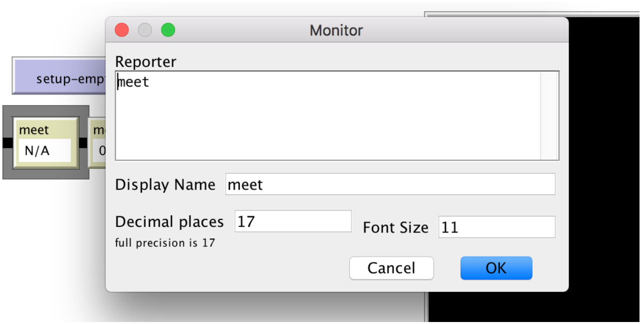
We can also define the **initialize-variables** procedure to call at the beginning of each run:



```
globals [
  ;; variables for each tick
  meet ;; how many people turtles meet (i.e., how many interactions there are)
  meet-same ;;  how many people turtles meet with same color
  meet-diff ;;  how many people turtles meet with different color
  coop-same ;;  how many interactions between coop-coop
  coop-diff ;;  how many interactions between coop-diff
  def-diff ;; how many interactions between diff-?

  ;; same variables as above but for the whole run
  total-meet
  total-meet-same
  total-meet-diff
  total-coop-same
  total-coop-diff
  total-def-diff
]

to initialize-variables
  ;; initialize all the variables
  set meet 0
  set meet-same 0
  set meet-diff 0
  set coop-same 0
  set coop-diff 0
  set def-diff 0
  set total-meet 0
  set total-meet-same 0
  set total-meet-diff 0
  set total-coop-same 0
  set total-coop-diff 0
  set total-def-diff 0
end
```

Interact - Monitors for the stats

We introduce few monitors and a plot to look at the number of interactions per each tick:



Monitor

Reporter
meet

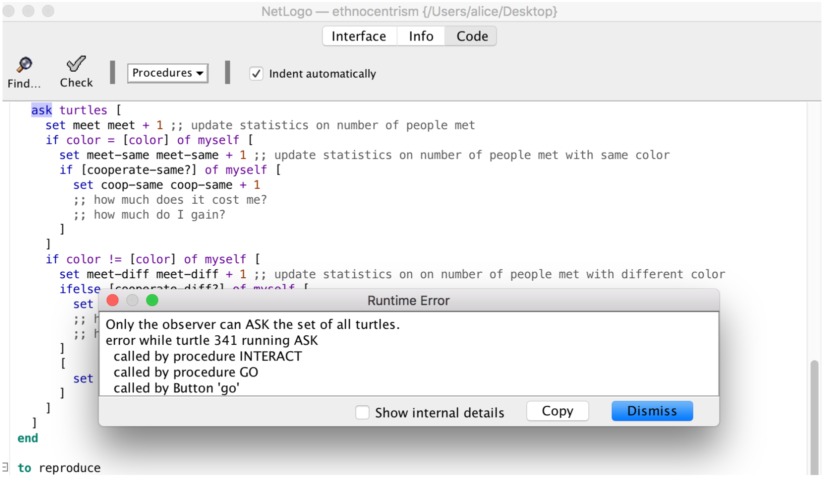
Display Name: meet

Decimal places: 17
full precision is 17

Font Size: 11

Interact - Error

Checking the code, we get the following error message:



```
  ask turtles [
    set meet meet + 1 ;; update statistics on number of people met
    if color = [color] of myself [
      set meet-same meet-same + 1 ;; update statistics on number of people met with same color
      if [cooperate-same?] of myself [
        set coop-same coop-same + 1
        ;; how much does it cost me?
        ;; how much do I gain?
      ]
    ]
    if color != [color] of myself [
      set meet-diff meet-diff + 1 ;; update statistics on on number of people met with different color
      ifelse [cooperate-diff?] of myself [
        set
        ;; h
        ;; h
      ]
      [
        set
      ]
    ]
  ]
end

to reproduce
```

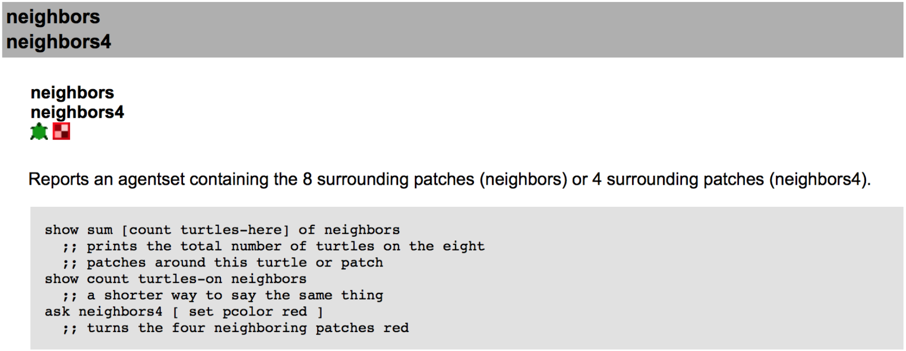
Runtime Error

```
Only the observer can ASK the set of all turtles.
error while turtle 341 running ASK
  called by procedure INTERACT
  called by procedure GO
  called by Button 'go'
```

We cannot ask the set of all turtles; moreover, it also makes sense that an agent interacts with only immigrants in its neighborhood.

Interact with neighbors

We can use the following command:



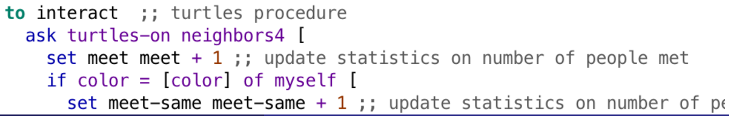
neighbors
neighbors4

neighbors
neighbors4

Reports an agentset containing the 8 surrounding patches (neighbors) or 4 surrounding patches (neighbors4).

```
show sum [count turtles-here] of neighbors
  ;; prints the total number of turtles on the eight
  ;; patches around this turtle or patch
show count turtles-on neighbors
  ;; a shorter way to say the same thing
ask neighbors4 [ set pcolor red ]
  ;; turns the four neighboring patches red
```

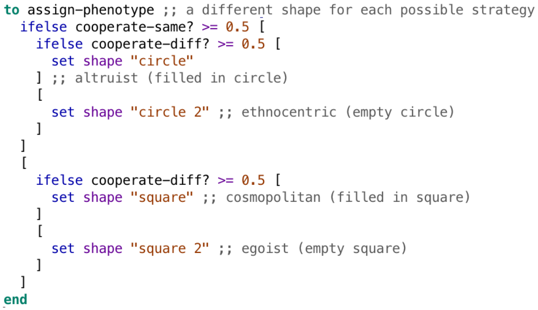
...and fix the **interact** procedure using **ask turtles-on neighbors**:



```
to interact  ;; turtles procedure
  ask turtles-on neighbors4 [
    set meet meet + 1 ;; update statistics on number of people met
    if color = [color] of myself [
      set meet-same meet-same + 1 ;; update statistics on number of pe
```

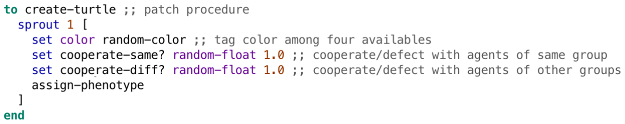
Four different shapes

We would like to be able to distinguish agents according to their strategy, thus we define the **assign-phenotype** procedure to change their shapes:



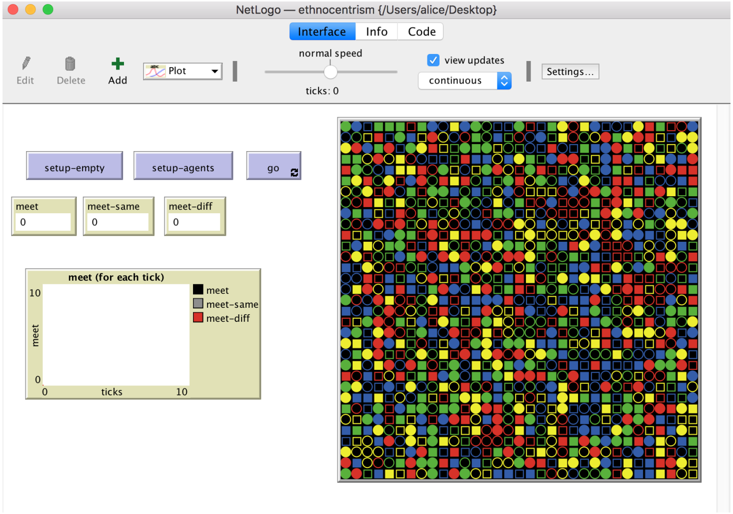
```
to assign-phenotype ;; a different shape for each possible strategy
  ifelse cooperate-same? >= 0.5 [
    ifelse cooperate-diff? >= 0.5 [
      set shape "circle"
    ] ;; altruist (filled in circle)
    [
      set shape "circle 2" ;; ethnocentric (empty circle)
    ]
  ]
  [
    ifelse cooperate-diff? >= 0.5 [
      set shape "square" ;; cosmopolitan (filled in square)
    ]
    [
      set shape "square 2" ;; egoist (empty square)
    ]
  ]
end
```

We call this procedure in **create-turtle**:



```
to create-turtle ;; patch procedure
  sprout 1 [
    set color random-color ;; tag color among four availables
    set cooperate-same? random-float 1.0 ;; cooperate/defect with agents of same group
    set cooperate-diff? random-float 1.0 ;; cooperate/defect with agents of other groups
    assign-phenotype
  ]
end
```

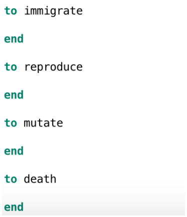
Our model - Version 0.2


NetLogo — ethnocentrism {/Users/alice/Desktop}

meet: 0
meet-same: 0
meet-diff: 0

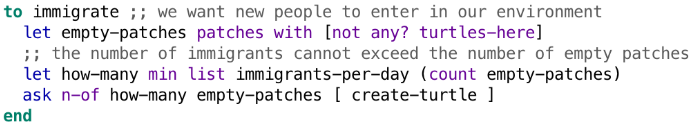
Next procedures to implement

We want our agents to interact with **immigrants**, to be able to **reproduce** and **die**, and their children may have the chance to **mutate**. In the following slides we'll define these procedures.



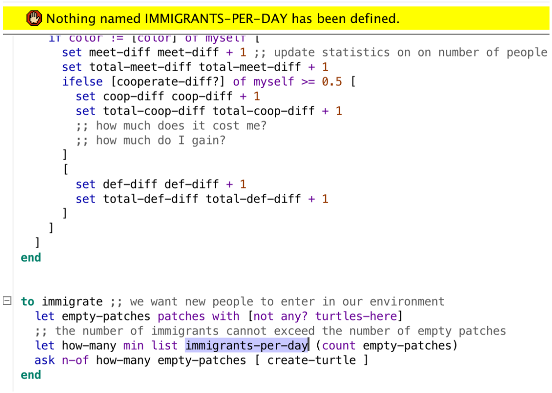
```
to immigrate
end
to reproduce
end
to mutate
end
to death
end
```

We talk about immigrants but we haven't defined yet a proper procedure to make them appear in the grid, and that's the purpose of **immigrate** below:



```
to immigrate ;; we want new people to enter in our environment
  let empty-patches patches with [not any? turtles-here]
  ;; the number of immigrants cannot exceed the number of empty patches
  let how-many min list immigrants-per-day (count empty-patches)
  ask n-of how-many empty-patches [ create-turtle ]
end
```

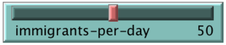
Immigrants-per-day

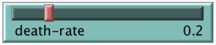


Nothing named IMMIGRANTS-PER-DAY has been defined.

```
    if color != [color] of myself [
      set meet-diff meet-diff + 1 ;; update statistics on on number of people
      set total-meet-diff total-meet-diff + 1
      ifelse [cooperate-diff?] of myself >= 0.5 [
        set coop-diff coop-diff + 1
        set total-coop-diff total-coop-diff + 1
        ;; how much does it cost me?
        ;; how much do I gain?
      ]
      [
        set def-diff def-diff + 1
        set total-def-diff total-def-diff + 1
      ]
    ]
  ]
end


to immigrate ;; we want new people to enter in our environment
  let empty-patches patches with [not any? turtles-here]
  ;; the number of immigrants cannot exceed the number of empty patches
  let how-many min list immigrants-per-day (count empty-patches)
  ask n-of how-many empty-patches [ create-turtle ]
end
```

To fix this, we add the following sliders in the Interface tab:





Reproduce

Each agent has its own **reproduction rate**, thus we add another turtles variable:

```
turtles-own [ reprod-rate cooperate-same? cooperate-diff? ] ;; agents have a reproduction rate and a strategy
```

Then we can write the **reproduce** procedure:

```
to reproduce ;;
  ;; if a random variable is bigger than reprod-rate, the agent can reproduce
  if random-float 1.0 > reprod-rate [
    ;; find an empty location to reproduce into, nearby the agent's actual position
    let destination one-of neighbors4 with [not any? turtles-here]
    if destination != nobody [
      hatch 1 [ ;; this allows to copy the current turtle,
        move-to destination ;; to move it to the destination patch
        mutate ;; and let the child mutate
      ]
    ]
  ]
end
```

Note: **hatch 1** allows to create a copy of the considered turtle...

Mutate

...but we don't want children to be exactly as their parents, thus we define the **mutate** procedure to allow them to apply a different strategy:

```
to mutate ;; children of agents can have different strategies from parents
  if random-float 1.0 < mutation-rate [
    let old-color color
    while [color = old-color]
      [ set color random-color ]
  ]
  ;; mutate the strategy flags;
  if random-float 1.0 < mutation-rate [
    set cooperate-same? 1 - cooperate-same?
  ]
  if random-float 1.0 < mutation-rate [
    set cooperate-diff? 1 - cooperate-diff?
  ]
  ;; fix the shape
  assign-phenotype
end
```

To do so, we add the **mutation-rate** slider in the Interface tab and randomly extract a value to change or not the strategy of the considered child; finally we update its phenotype accordingly.

Death

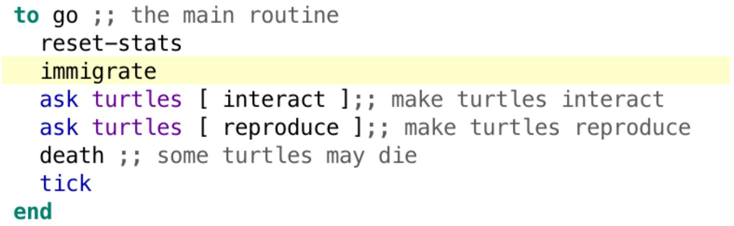
Unfortunately, each agent has also a chance of dying (and being removed from the grid):

```
to death
  ;; check to see if a random variable is less than the death rate for each agent
  ask turtles [
    if random-float 1.0 < death-rate [ die ]
  ]
end
```

Similarly to mutation, we introduce a **death-rate** slider in the Interface tab.

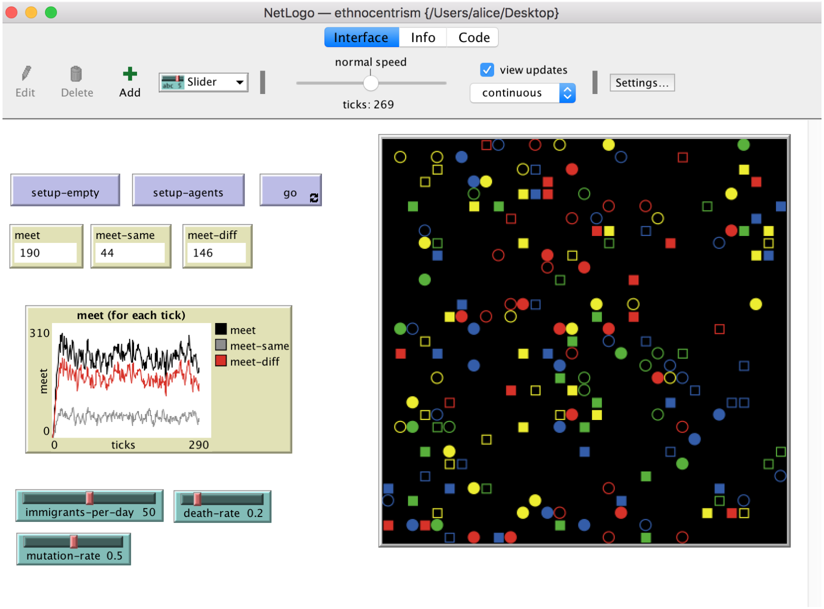
Update go

We update the go procedure, in order to apply these improvements in our model:



```
to go ;; the main routine
  reset-stats
  immigrate
  ask turtles [ interact ];; make turtles interact
  ask turtles [ reproduce ];; make turtles reproduce
  death ;; some turtles may die
  tick
end
```

Our model - Version 0.3


NetLogo — ethnocentrism {/Users/alice/Desktop}

Interface | Info | Code

normal speed

ticks: 269

view updates: continuous

meet 190

meet-same 44

meet-diff 146

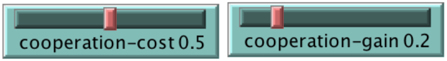
immigrants-per-day 50

death-rate 0.2

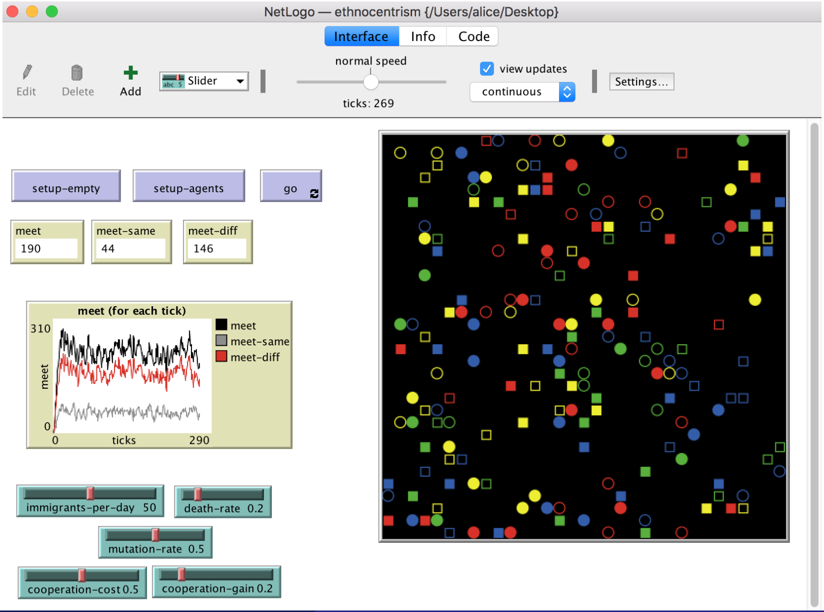
mutation-rate 0.5

Cooperation cost and gain

We improve our **interact** procedure, assuming that there are a cost and a gain when two people interact, thus we add two sliders in the Interface tab and we update the code:


cooperation-cost 0.5
cooperation-gain 0.2

Our model - Version 0.4



NetLogo — ethnocentrism {/Users/alice/Desktop}

normal speed

ticks: 269

view updates ✓

continuous

meet 190

meet-same 44

meet-diff 146

immigrants-per-day 50

death-rate 0.2

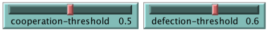
mutation-rate 0.5

cooperation-cost 0.5

cooperation-gain 0.2

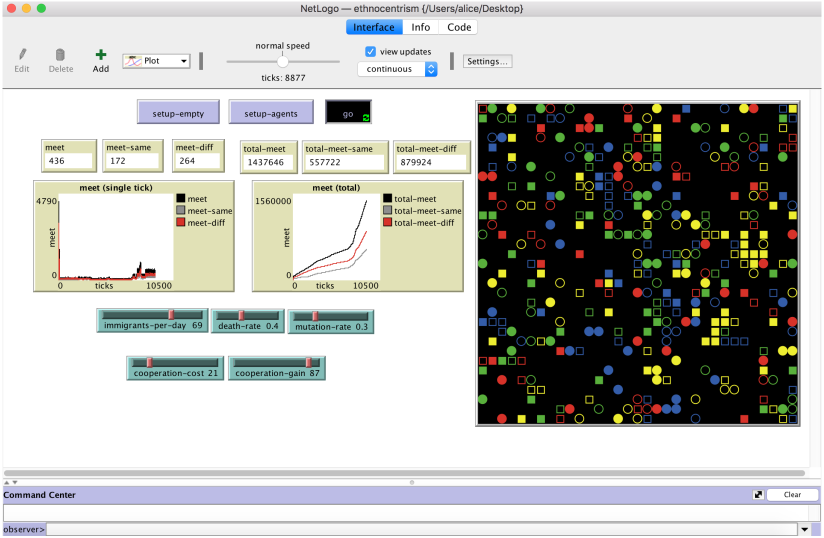
Interact - Cooperation/defection thresholds

Another small improvement: instead of comparing an agent's *cooperation-same?* and *cooperation-diff?* with 0.5, we introduce two thresholds with two sliders:


cooperation-threshold 0.5
defection-threshold 0.6

```
to interact  ;; turtles procedure
  ask turtles-on neighbors4 [
    set meet meet + 1 ;; update statistics on number of people met
    set total-meet total-meet + 1
    if color = [color] of myself [
      set meet-same meet-same + 1 ;; update statistics on number of people met with same color
      set total-meet-same total-meet-same + 1
      if [cooperate-same?] of myself >= cooperation-threshold [
        set coop-same coop-same + 1
        set total-coop-same total-coop-same + 1
        ;; how much does it cost to an agent to cooperate?
        ask myself [ set reprod-rate reprod-rate - cooperation-cost ]
        ;; how much does it gain?
        set reprod-rate reprod-rate + cooperation-gain
      ]
    ]
    if color != [color] of myself [
      set meet-diff meet-diff + 1 ;; update statistics on on number of people met with different color
      set total-meet-diff total-meet-diff + 1
      ifelse [cooperate-diff?] of myself >= defection-threshold [
        set coop-diff coop-diff + 1
        set total-coop-diff total-coop-diff + 1
        ;; how much does it cost to an agent to cooperate?
        ask myself [ set reprod-rate reprod-rate - cooperation-cost ]
        ;; how much does it gain?
        set reprod-rate reprod-rate + cooperation-gain
      ]
      [
        set def-diff def-diff + 1
        set total-def-diff total-def-diff + 1
      ]
    ]
```

Our model - Version 0.5



NetLogo — ethnocentrism {/Users/alice/Desktop}

Interface | Info | Code

normal speed

ticks: 8877

view updates: continuous

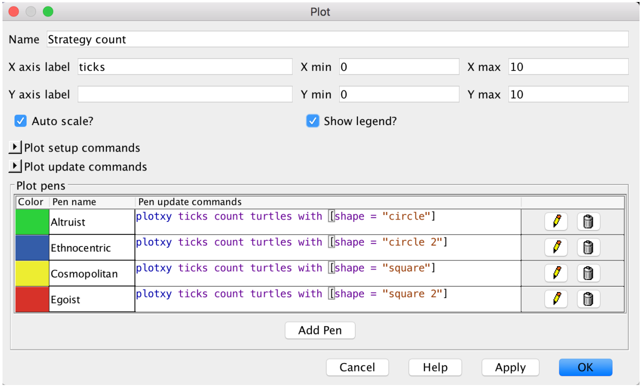
meet: 436
meet-same: 172
meet-diff: 264

total-meet: 1437646
total-meet-same: 557722
total-meet-diff: 879924

immigrants-per-day 69
death-rate 0.4
mutation-rate 0.3
cooperation-cost 21
cooperation-gain 87

Strategy counts

It would be nice to see if, changing all these values and parameters during a run, the number of agents having a certain strategy changes or not. Let's introduce a plot to represent all strategies, counting the number of agents with a given shape:



Name: Strategy count

X axis label: ticks — X min: 0 — X max: 10

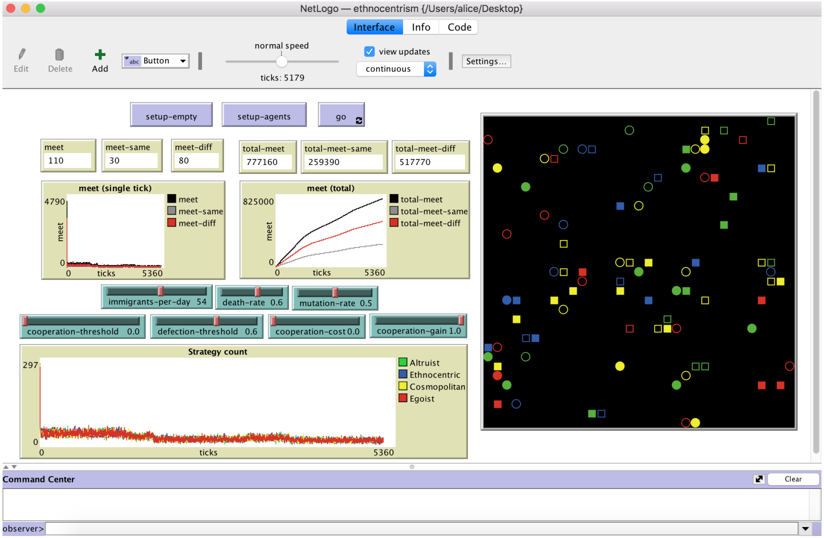
Y axis label: — Y min: 0 — Y max: 10

[✓] Auto scale? [✓] Show legend?

Plot pens

Color	Pen name	Pen update commands
🟩	Altruist	`plotxy ticks count turtles with [shape = "circle"]`
🟦	Ethnocentric	`plotxy ticks count turtles with [shape = "circle 2"]`
🟨	Cosmopolitan	`plotxy ticks count turtles with [shape = "square"]`
🟥	Egoist	`plotxy ticks count turtles with [shape = "square 2"]`

Our model - Version 1


NetLogo — ethnocentrism {/Users/alice/Desktop}

Interface Info Code

normal speed

ticks: 5179

view updates: continuous

setup-empty setup-agents go

meet 110

meet-same 30

meet-diff 80

total-meet 777160

total-meet-same 259390

total-meet-diff 517770

immigrants-per-day 54

death-rate 0.6

mutation-rate 0.5

cooperation-threshold 0.0

defection-threshold 0.6

cooperation-cost 0.0

cooperation-gain 1.0

What to bring home

- We have done a sort of **reverse engineering** of a model already present in NetLogo, building it piece by piece, understanding how it has been designed and which steps were needed to realize it;
- There are still **few differences** between our model and Hammond and Axelrod's one: you can try to find them reading the Info tab of their model;
- You can also study **other similar NetLogo models**: *Altruism*, *Cooperation Segregation*, etc.;
- We have seen that NetLogo can be a powerful tool in several fields, and Social Science is just one of them: free your curiosity and **explore the Models Library**! Let it inspire you, maybe you'll decide what your next model could be about...

References

- Hammond, R.A. and Axelrod, R. (2006). *The Evolution of Ethnocentrism*. doi:10.1177/0022002706293470. Journal of Conflict Resolution 50, no. 6, pag. 926–36.
- Wilensky, U. (2003). NetLogo Ethnocentrism model. http://ccl.northwestern.edu/netlogo/models/Ethnocentrism. Center for Connected Learning and Computer-Based Modeling, Northwestern University, Evanston, IL.