Agent-Based Modelling in NetLogo

Lists and El Farol Bar model

David Hales

www.davidhales.com/abm-netlogo

Lists in NetLogo

- Lists are a sequence of items
- Items can be any type (including lists)
- Creating lists:

```
let a [ 1 2 3 4 ] ; list of 4 numbers
let b [ [ 1 2 ] [ 3 4 ] ] ; list of 2 lists each 2 numbers
let c [ “hello” 1 ] ; string, number
let d list turtle 0 turtle 1 ; list of two turtles
let d list turtles patches ; list of two agentsets
```

Lists in NetLogo

- n-values generates new lists by calling a reporter iteratively:

```
let a n-values 5 [?]
let b n-values 7 [random 10]
print a
print b
```

- Would produce something like:

```
[0 1 2 3 4]
[0 5 3 6 3 9 4]
```

Lists in NetLogo

- Adding items to the start and end of a list:

```
let a [1 2 3 4]
let b fput 0 a    ; b is a with added 0 at front
let c lput 0 a    ; c is a with added 0 at end
print b print c
```

- Would display:

  [0 1 2 3 4]

  [1 2 3 4 0]

Lists in NetLogo

- Removing items from front and end of a list:

```
let a [0 1 2 3 4]
let b but-first a   ; b is a without first item
let c but-last a    ; c is a without last item
print b print c
```

- Would display:

```
[1 2 3 4]
[0 1 2 3]
```

Lists in NetLogo

- Retrieving individual items from a list:

```
let a [ “world” 1 2 “hello” 4 [5 6] ]
print first a
print last a
print item 3 a
```

- Would display:

```
world
[5 6]
hello
```

Lists in NetLogo

- Retrieving subsists from a list:

  let a [“this” “is” 2 3 “it”]

  print sublist a 0 2   ; would display: [“this” “is”]

  print sublist a 2 4   ; would display: [2 3]

- Checking if item is member of a list:

  print member? “this” a   ; would display: true

  print member? 4 a   ; would display: false

Lists in NetLogo

- map applies a reporter to each item in a list or multiple lists:

```
let a [0 1 2 3 4]
print map [? * 2] a
print map [? = “hi”] [1 2 “hi” “there”]
print (map [?1 * ?2] [1 2 3] [2 1 5]) ; two lists
```

- Would display:

```
[0 2 4 6 8]
[false false true false]
[2 2 15]
```

Lists in NetLogo

- foreach iterates over items in a list, executing a command block:

```
let c 0
foreach [1 2 3] [set c c + ?]
create-turtles c
let a (list turtle 0 turtle 1 turtle 2)
foreach a [ask ? [forward 1]]
```

- Would create 6 turtles and move the first 3 forward 1

Lists - Task 1

- Write a program in NetLogo that:
  - creates N turtles and places them randomly
  - each turtle stores an initial empty list
  - turtles move around in a random walk
  - If a turtle meets another they have not met before they add their “who” number to end of their list
    - A turtle meets another if it shares the same patch
  - When a turtle has met all other turtles it stops moving
  - When all turtles have stopped the program stops

Task 1 – One way of doing it

```
turtles-own [ met ]

to setup
 clear-all
 create-turtles N [
  setxy random-xcor random-ycor ; place randomly
  set met [] ; empty list
 ]
 reset-ticks
end

to go
 let moved 0
 ask turtles [
   if length met < N - 1 [ ; if not met all turtles
     set moved moved + 1
     forward random 4 rt random 360 ; random walk
     foreach [who] of other turtles-here [
       if not (member? ? met) [set met lput ? met]
     ]
   ]
 ]
 if moved = 0 [stop]  ; if nobody moved then stop
end
```

El Farol Bar Model

“Nobody goes there anymore. It's too crowded” (Yogi Berra)

El Farol bar

- There is a really good bar called El Farol Bar
- But when it's busy it's not much fun
- Each Thursday agents make a decision:
  - Go to bar OR stay at home
- If T > 60% of the agents go to bar it's busy
- If it's busy it would be better to stay at home
- Hence agents have a preference ordering of:
  - Attend non-busy > stay at home > attend busy
- Problem: all make decision at once, no communication, too late to go home if busy

W. Brian Arthur (1994) "Inductive Reasoning and Bounded Rationality", American Economic Review, 84

El Farol bar

- Arthur wishes to examine what happens when agents use induction to make decisions:
  - Agents know past attendance history at bar
  - Use simple rules to attempt to predict attendance
  - Each agent stores several such “predictor” rules
  - Map past attendance figures to next attendance
  - Uses the currently best predictor it has based on how well it would have worked predicting past attendance
- This can be contrasted with traditional economic deduction and game theory which looks for equilibrium points

El Farol bar

- Examples of predictors could be:
  - Same number as last week
  - Same number as 3 weeks ago
  - Average of last 4 weeks
- Arthur notes no single predictor could work best for everyone since:
  - If all predicted same all would lose
- Hence any pattern that emerged in attendance would tend to disappear

What does Arthur find?

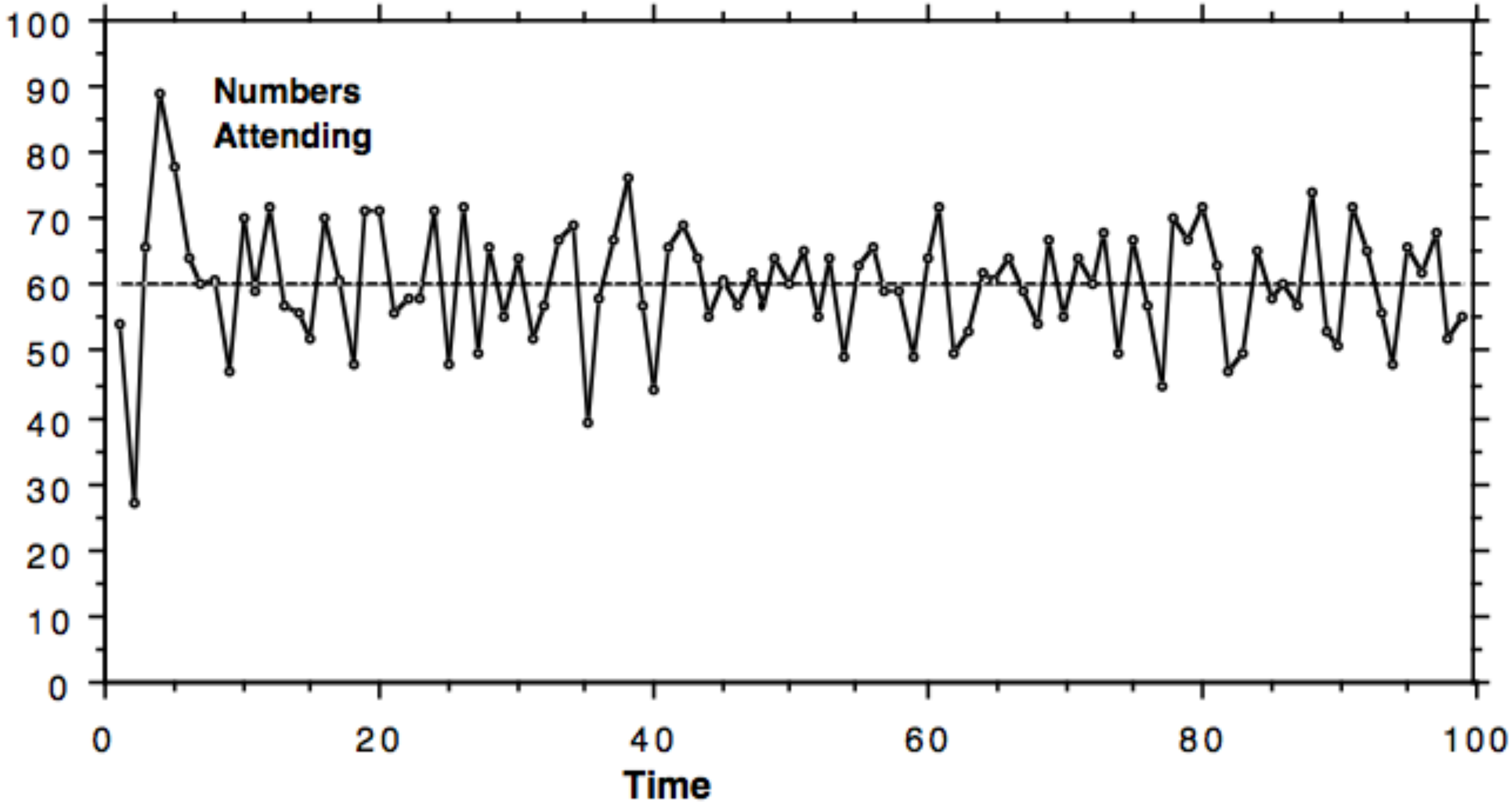

FIGURE 1. BAR ATTENDANCE IN THE FIRST 100 WEEKS.

Observations

- Arthur noticed attendance tends to average around 60%
- But that a dynamic “ecology” of active predictors causes this
  - Agents constantly changing which of their predictors is the best based on history
- This is because when active predictors predicted wrongly then others become active

The NetLogo El Farol Bar model

- NetLogo provides an implementation of Brian Arthurs El Farol Bar model:
  - Sample models > social science > El Farol
- It uses lists to implement agents with simple inductive learning abilities
- The exercise sheet gives a detailed description of the code and some tasks
- Here I presented a *slightly* modified version

El Farol model

- There are 100 agents (turtles), bar is crowded if > 60 attend (attendance threshold)
- There is a list called *history* that stores previous attendance numbers at the bar
- Each agent stores a list of *strategies*
- Each strategy produces a prediction of the next attendance based on past history
- Each agent keeps track of the current best predicting strategy – used for next prediction

El Farol model

- Each tick of the model represents one week
  - Each agent determines its best strategy based on the history
  - Each agent uses its best strategy to make a prediction of how many agents will attend next
  - It uses this prediction to decide if to attend the bar or not
  - The actual attendance that week is then added to the history list

El Farol model

- Assuming memory size (m) = 2
- The history is a list of attendances of size m*2. First item (h0) is most recent attendance value:
  - history = [h0 h1 h2 h3 h4]
  - e.g. [30 20 5 90]
- A strategy is a list comprising a constant followed by m weights. Constant and weights are random values between -1 and +1:
  - strategy = [c w1 w2]
  - e.g. [0.5 -0.1 0.6]

El Farol model

- To make a prediction a strategy takes the weighted sum of the last m attendance values from the history and adds the constant*100 (the number of agents)
- Note: these strategies are quite dumb they could predict < 0 or > 100 attendance!

El Farol model

- Given a history h and strategy s:

  where: m=2, h = [h0 h1] and s = [c w1 w2], then:

  prediction = h0*w1 + h1*w2 + c*100

- e.g.: h = [30 20], s = [0.5 -0.1 0.6]

  = 30*-0.1 + 20*0.6 + 0.5*100

  = -3 + 12 + 50

  = 59 (predicted next attendance)

El Farol model

- Hence the following strategies would predict:
  - [0 0 0] always zero
  - [0.7 0 0] always 70
  - [0 1 0] the same as last week
  - [0 0 1] the same as the week before last
  - [0 0.5 0.5] average of the last two weeks
  - [0 0.9 -0.1] 90% of last week – 10% of week before
  - [-0.1 0.5 0.5] average of last two weeks – 10
  - [-1 0 0] always -100

NetLogo – predict attendance

```
; strategy is a single strategy (a list)
; subhistory is a history of length memory-size (a list)

to-report predict-attendance [strategy subhistory]
  report first strategy * 100 + sum (map [?1 * ?2] butfirst strategy subhistory)
end
```

Note: NetLogo library version omits the *100 of the constant

El Farol model

- Each agent stores a list of ns strategies. For example if ns = 2 then:
  - e.g. Strategies = [[0 0.5 0.5] [0.5 -0.1 0.6]]
- Each agent could take any of its ns strategies and produce a prediction for the next attendance
- But which one should it use? Which is the “best strategy”?

El Farol model

- To find the best strategy an agent goes back in history and evaluates how well each strategy would have done if it had been used to predict the next attendance
- It goes back week-by-week (for m weeks) evaluating how well each strategy would have predicted each of those weeks => **score**
- It then selects the strategy with the best score as the current best strategy and uses this for predicting the next new week

El Farol model

- A strategy score is calculated as the sum of differences between its predictions and the actual attendances i.e. the sum of errors
- Hence low score is good, high score is bad
  - Best score = 0 (indicates perfect prediction)
  - Worst score = 800 (given 100 agents, m=2, then worst possible score = 100*(m+2)*m = 800)

Example calculating a score for each strategy based on history for memory size (m) = 2 and number of strategies (ns) = 2

History

h0	h1	h2	h3
30	20	5	90

Strategy 1

c	w1	w2
0	0.5	0.5

Strategy 2

c	w1	w2
0.5	-0.1	0.6

To calculate score (error):
p0 = predict h0 using h1, h2
p1 = predict h1 using h2, h3
error = | h0 – p0 | + | h1 – p1 |

To predict using h1, h2:
prediction = w1*h1 + w2*h2 + c*100

Strategy 1 score:
p0 = 20*0.5 + 5*0.5 + 0*100 = 12.5
p1 = 5*0.5 + 90*0.5 + 0*100 = 47.5
error = | 30 – 12.5 | + | 20 – 47.5 |
= 17.5 + 27.5
= 45

Strategy 2 score:
p0 = 20*-0.1 + 5*0.6 + 0.5*100 = 51
p1 = 5*-0.1 + 90*0.6 + 0.5*100 = 47.5 = 104
error = | 30 – 51 | + | 20 – 104 |
= 21 + 84
= 105

Strategy 1 is best because score (error) is lower

NetLogo – find best strategy

```
; a turtle evaluates each of its strategies by calculating a score for each
; based on how well it predicts past history and then sets best-strategy to
; the one with the best (lowest) score
to update-strategies
  ;; initialize best-score to a maximum, which is the worst possible score
  let best-score 100 * (memory-size + 2) * memory-size
  foreach strategies [
    let score 0
    let week 1
    repeat memory-size [
      set prediction predict-attendance ? sublist history week (week + memory-size)
      set score score + abs (item (week - 1) history - prediction)
      set week week + 1
    ]
    if (score <= best-score) [
      set best-score score
      set best-strategy ?
    ]
  ]
end
```

Note: NetLogo library version initialises best-score to a different value

NetLogo – main loop

```
to go
  ;; each agent predicts attendance at the bar and decides whether or not to go
  ask turtles [
    set prediction predict-attendance best-strategy sublist history 0 memory-size
    set attend? (prediction < overcrowding-threshold)  ;; true or false
  ]
  ;; depending on their decision, the agents go to the bar or stay at home
  set attendance count turtles with [attend?]
  ;; update the attendance history
  ;; remove oldest attendance and prepend latest attendance
  set history fput attendance but-last history
  ;; the agents decide what the new best strategy is
  ask turtles [ update-strategies ]
  ;; advance the clock
  tick
end
```

What has this got to do with economics?

- If view go to bar = sell, stay at home = buy (or vice versa)
- If everyone is selling price goes down
- If everyone is buying price goes up
- To "win" (buy low, sell high) you need to:
  - Buy when everyone is selling
  - Sell when everyone is buying
  - Be in a minority!
- Everyone is predicting what will happen next
- Traditional economic models look for equilibrium states (stabilities) but these kinds of models look at non-equilibrium outcomes

Aside: Nash equilibrium

- In fact there is a so-called Nash equilibrium for El Farol if agents can use "mixed strategies"
  - pure strategy = move in the game (go to bar or stay at one)
  - mixed strategy = probability dist. over moves (go to bar with some probability)
- Nash eq. everyone go to bar with prob(0.6)

Minority Game

- A simplification of El Farol bar
- N agents (odd number), One of two actions (0 or 1)
- Those who choose minority action win
- Agents store:
  - History: M last outcomes 0 or 1 (which action in minority)
  - Some number of predictors map M last outcomes to predicted next outcome that will be in minority (0 or 1)
- This formulation used to:
  - apply various forms of analysis
  - modified to produce outcomes that capture "stylised facts" (statistical regularities) found in real markets

Review paper: Tobias Galla, et al (2006) Anomalous fluctuations in Minority Games and related multi-agent models of financial markets. arXiv:physics/0608091v1

NetLogo model library / sample models / social science / Minority Game

Non-equ. Market models

- Good blog post on minority game by Mark Buchanon:
  - http://physicsoffinance.blogspot.hu/2012/02/minority-games.html
- Paper: B. LeBaron (2002) “Building the Santa Fe Artificial Stock Market,” Working Paper, Brandeis University
  - More sophisticated model capturing various aspects of a stock market
- Brian Arthur at World Economic Forum talking about complexity economics:
  - https://www.youtube.com/watch?v=Lx-pRkp7pM8
- Good lecture by Brian Arthur on economics (history) technology development and algorithmic approaches:
  - https://www.youtube.com/watch?v=WQ6ppznYl-Q