

Netlogo 워크숍

이남형¹

¹연세대학교 경영연구소

2017.10.27.

- 1 서론
- 2 Netlogo 시작하기
- 3 나비모형: ODD 프로토콜
- 4 나비모형: Netlogo
- 5 나비모형: 과학 연구를 향하여

- ◀ ◻ ▶ ◀ ◻ ▶ ◀ ≡ ▶ ◀ ≡ ▶ ≡

개요

- 목표
 - 행위자 기반 모형의 기본적인 특성을 이해하고, Netlogo 프로그래밍의 기초를 습득함
- 교재 및 일정
 - 10월 27일, Steven F. Railsback and Volker Grimm (2012), *Agent-Based and Individual-Based Modelling: A Practical Introduction*.
 - 행위자 기반 모형은 무엇인가: Ch. 1.
 - Netlogo 기초: Ch. 2.
 - 나비 모형: Ch. 3-5.
 - 11월 3일, Lynne Hamill and Nigel Gilbert (2016), *Agent-Based Modelling in Economics*.
 - 쇼핑 모형: Ch. 2.
 - 11월 10일, Hamill and Gilbert (2016)
 - 신기술 확산(전화 보급) 모형: Ch. 4.
- 장소 및 시간
 - 연세대학교 경영관 B225, 오후 4시 - 오후 6시

행위자는 무엇인가?

- [Page, 2008]

Agent-based models consist not of real people but of computational objects that interact according to rules.

- 행위자
 - 조직, 인간, 제도 등 어떤 목적을 갖고 있기만 하면 됨

모형은 무엇인가?

- [Starfield et al., 1990]

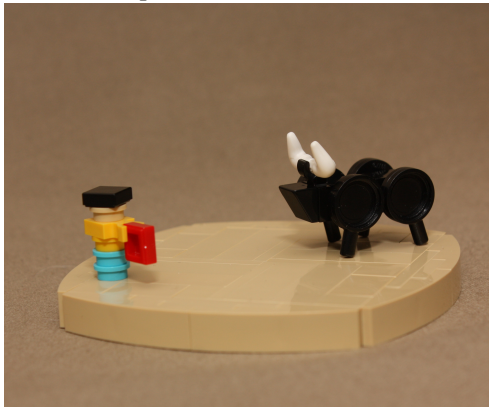
A model is a purposeful representation of some real system.

- 모형의 사용 목적
 - 사물이 어떻게 작동하는 지 이해하기 위해;
 - 관찰된 패턴을 설명하기 위해;
 - 변화에 시스템의 변화가 어떻게 반응하는 지 예측하기 위해

모형은 현실의 어디까지를 반영해야 하는가?

출처: [https:](https://www.flickr.com/photos/26629915@N03/36879851166/)

[//www.flickr.com/photos/26629915@N03/36879851166/](https://www.flickr.com/photos/26629915@N03/36879851166/)



- 모형의 구체적 목적에 따라
 - 모형을 사용하여 우리가 궁금해하는 질문에 대한 답 → 필터로 작동
 - 예) 지도
 - 즉, 질문에 답하는 데 상관이 없거나 중요성이 충분하지 않은 현실의 특징은 제외시켜야 함

- 모형을 정식화 하기 위해 기초적인 아이디어가 필요
 - 어떻게 시스템이 작동하는가에 대한 경험적 지식
 - 또는 유사한 질문을 다룬 기존 모형
 - 또는 이론
 - 아니면 상상
- 그리고 우리는 모형이 실재의 세계를 얼마나 잘 반영하고 있는 지 평가할 기준이 필요
 - 기준은 실재 시스템을 식별하고 특징 지을 수 있는 패턴이나 규칙성에 기반해야 함
- 그리고 모형에 대한 평가 이후 모형을 수정 → 재평가 → 재수정 ...

행위자 기반 모형은 무엇이 다른가?

- 행위자 기반 모형
 - 계산 가능성의 확장
 - 고유성/이질성 부여 가능
 - 행위자의 크기, 위치, 보유 자원, 과거 기록 등이 각기 다르게 함
 - (국지적) 상호작용
 - 행위자는 다른 행위자와 상호작용하지만, 그들의 이웃하고만 상호작용
 - 행위 규칙
 - 모형의 목적에 따라 결정

행위자 기반 모형의 장점

- 장점
 - 적응적 행동
 - 행위자는 그 자신, 다른 행위자 그리고 환경의 현재 상태에 행동을 적응시켜 간다.
 - 창발
 - 시스템의 개별 구성 요소가 상호 작용 및 환경에 반응하면서 시스템 동학을 만든다.
 - 분석 층위를 넘나들
 - 시스템의 개체로 인해 시스템 차원의 사건이 발생
 - 시스템 차원의 사건으로 인해 개체도 영향을 받음

행위자 기반 모형 대표 연구 및 온라인 사이트

- 대표 연구(경영)
 - [Bonabeau, 2002]
 - [Rand and Rust, 2011]
- 행위자 기반 모형 소개(다소 오래됨)
 - <http://www2.econ.iastate.edu/tesfatsi/abusiness.htm>
- 온라인 강좌
 - <https://www.complexityexplorer.org>
- 모형 소스코드
 - <https://www.openabm.org>
 - <http://modelingcommons.org/account/login>

Netlogo 훑어보기

- Netlogo 설치
 - <https://ccl.northwestern.edu/netlogo/download.shtml>
- 매뉴얼과 친해지기
 - <https://ccl.northwestern.edu/netlogo/docs/>

행위자와 변수

- 행위자
 - Mobile agents → turtles.
 - Patches → the space (the square cells).
 - Links → connecting turtles.
 - The observer → an overall controller of a model.
- 변수
 - 행위자별 내장 변수
 - Turtles: breed, color, heading, hidden? ...
 - Patches: pcolor, plabel, ...
 - Links: breed, color, end1, end2, ...
 - 각 행위자별로 변수를 추가할 수 있음
 - the observer 변수는 자동으로 전역 변수(global variables) → 모든 행위자가 읽거나 쓸 수 있는 변수.

- 행위자는 자신의 타입이 사용할 수 있는 변수에만 접근할 수 있음
 - 즉, turtles의 변수에 patches가 접근할 수 없음
- 예외
 - observer 변수는 전역 변수로서 모든 agent가 사용 가능
 - turtles은 자신이 현재 올라서 있는 patch의 변수를 자동으로 사용 가능

A Primitive

- A Primitive
 - 행위자가 할 것을 지시하는 내장된 절차 또는 명령 →
노가다를 줄이려면 사건을 열심히 봐둘 것
 - commands: 행위자에게 행동을 지시
 - reporters: 값을 계산하고 이를 사용할 수 있도록 보고
 - 맥락이 있음
 - a turtle, patch, link, or observer 가 사용할 수 있는 primitive가
정해져 있음
 - uphill은 turtle 만; dictionary에 icon으로 표시됨
 - 오류 메시지: “using a patch command in a turtle context.”

ODD 프로토콜

- 많은 경우 ABM을 reimplement 하거나 결과를 replicate하기 어려움
- 모델에 대한 묘사가 사실 설명, 정당화, 기타 등등 것들에 대한 논의를 다 담아야 하기 때문
- 손 쉽게 이해할 수 있도록 표준화된 설명 방식을 만들면 되지 않을까? → 프로토콜
- `ODD_protocol.pdf/xls` 참고

나비 모형과 ODD 프로토콜

- 목적
 - 나비의 언덕 오르기 행위와 지형도의 상호 작용으로 통로가 만들어지는 조건은 무엇인가?
 - 나비의 언덕 오르기에 영향을 미치는 다양한 요소가 통로의 창발에 영향을 미치는가?

- 독립체, 상태 변수, 비례
 - 나비
 - 위치가 중요 특징 → patch의 중앙에 위치 시킴
 - patch
 - 150×150
 - patch는 하나의 변수만 가짐: elevation
 - 1,000 steps 시행, a patch는 $25 \times 25m^2$ 에 대응

- 전체 과정 및 스케줄
 - 과정: 단일. 나비의 움직임
 - 매 step마다 나비가 한 번 움직임
 - 기본 모형에서는 상호작용이 없으므로, 나비의 움직임 순서는 중요하지 않음

- 초기화
 - 지형도
 - 단순 가상 공간에서
 - 실제 공간으로
 - 나비 500마리 → 하나의 patch 또는 좁은 지역에 위치
- 입력 자료
 - 환경 변화 없음 → 입력 자료 없음
- 하위 모델
 - 나비가 어떻게 언덕을 오르는가? 곤장 또는 무작위로?

ODD로부터 Netlogo로

- 새로운 파일: Butterfly-1.nlogo
- 목적
 - <http://www.railsback-grimm-abm-book.com/Chapter04/ButterflyModelODD.txt>
 - Info 탭에 작성

- 독립체, 상태 변수, 비례

```
globals
```

```
[  
]
```

```
patches-own
```

```
[  
]
```

```
turtles-own
```

```
[  
]
```

- Check 클릭 → 아무 문제 없어야 정상
- ODD를 보면,
 - turtles의 상태 변수는 위치 → 따로 정할 필요 없음
 - patches의 상태 변수 elevation → 정해야 함

```
patches-own  
[  
elevation  
]
```

- now, move on.
- 모형의 공간
 - Interface tab → Settings.
 - Location of origin → Corner and Bottom Left.
 - max-pxcor 와 max-pycor → 149
 - World wraps horizontally와 World wraps vertically
체크 해제.
 - World 창이 너무 크게 보일 것, 수정해봅시다.
 - Interface tab → Settings.
 - Patch size → 3.

- 초기화

```
turtles-own  
[  
]
```

```
to setup  
  ca  
  ask patches  
  [  
  
  ]  
  reset-ticks  
end
```

- 지형 생성

```
ask patches
[
  let elev1 100 - distancexy 30 30
  let elev2 50 - distancexy 120 100

  ifelse elev1 > elev2
  [set elevation elev1]
  [set elevation elev2]

  set pcolor scale-color green elevation 0 100
]
```

Note Netlogo의 명령어는 의미별로 모두 띄어쓸 것

- A set of small navigation icons typically found in Beamer presentations, including symbols for back, forward, search, and other slide controls.

- 움직이는 행위자 turtle을 만들자.

```
set pcolor scale-color green elevation 0 100  
]
```

```
crt 1  
[  
  set size 2  
  setxy 85 95  
]
```

- 잘 만들어졌나 확인

- 스케줄 → 나비(turtle)가 1 step에 1번 움직임

```
  reset-ticks  
end
```

```
to go  
  ask turtles [move]  
end
```

```
to move
```

```
end
```

- → 1000 steps

```
to go
  ask turtles [move]
  tick
  if ticks >= 1000 [stop]
end
```

- Interface에 go button 추가
 - Interface에서 구성을 움직이거나, 리사이즈, 편집하고 싶다면,
 - 구성 요소를 선택하고 마우스 오른쪽 클릭 후, "Select" 선택
- go 눌러 보시다 → 아무 일도 없어야 정상

- to move의 내용을 채우지 않았음
- q 의 확률로 곧장 언덕을 오르고, $1 - q$ 의 확률로 무작위 이동
→ 국지적인 정상에 오를 때까지

```
to move
  ifelse random-float 1 < q
    [uphill elevation]
    [move-to one-of neighbors]
end
```

- to move
 - random-float NUMBER → manual 참고
 - uphill, move-to one-of neighbors: 모두 primitives
- check or go → 에러!, q 정의 필요

```
globals
[
  q
]
...
to setup
...
  set q 0.4
end
```

- 나비들은 잘 가고 있을까?

```
crt 1  
[  
  set size 2  
  setxy 85 95  
  pen-down  
]
```

- 한번에 보내고 싶다면
 - Interface 탭, go 버튼, 오른쪽 클릭, forever 박스 체크

논문에는 안 나오는 것

- 진짜 일은 최초의 모델이 만들어진 이후에 시작
 - 모형 분석과 세부 수정의 반복 작업
 - → 모형을 갖고 놀기 (Playing)
 - → “What would happen if ...”의 마인드가 중요
- NetLogo 모델 라이브러리
 - 실현에는 장점 → 그들이 세운 가정과 수식이 어떻게 작동하는지 보여줌
 - 그러나 이 것을 가지고 어떻게 과학을 할지 알려주지 않음
 - 하지만 어떻게 아이디어와 개념을 만들었고,
 - 가설을 발전시키고 테스트 했는지,
 - 관찰된 현상을 얼마나 좁게 또는 얼마나 일반화시켜서 설명할 것인지 등을 보여주지 않음

나비 모형으로 과학하기

- 나비 모형의 목적: 통로
 - 즉, 나비를 유인하는 어떤 요소가 없는 데도, 나비가 움직이는 경로가 집중된다는 것
 - 그리고 이 것이 어떻게 언제 나타나는 지는 설명해야함
 - 하지만 현재의 모델은 나비의 움직임만 모델링
 - 가상 지형을 실제 지형으로 대체

- 무엇을 할 지부터 생각해야 함
- 통로의 정의가 필요
 - 우리 모델에서 나비는 아무데서나 출발하고 멈출 수 있음
 - 나비가 국지적 정상에 도달하면 멈춘다고 가정
 - 국지적 정상: 주변의 8개 patch보다 높은 patch
 - 모든 나비가 사용하는 통로의 폭

$$= \frac{\text{각각의 나비가 방문하는 patch}}{\text{출발점과 도착점을 연결하는 (평균) 직선 거리}}$$

- 모든 나비가 같은 직선 경로를 따라 올라간다고 생각하면, 통로의 폭은 작을 것 → 다른 경로를 따른다면, 폭이 커질 것
- 모델의 분석을 위해 통로의 폭과 나비가 곧장 언덕을 오를 확률 q 의 관계 그래프를 그려보자.

프로그램의 수정

- Interface 탭, sliders 생성 $\rightarrow q$ 변경
 - 0.0에서 1.0까지 0.01씩 증가하도록
- 경고!
 - sliders, switches, choosers는 모두 전역 변수를 사용
 - 따라서 중복 \rightarrow 코드에서 주석 처리
 - \rightarrow 관련 명령도 모두 주석 처리
 - `set q 0.4`를 그대로 두면 $q = 0.4$ 로 고정

주석

- ;
 - 지금 하고 있는 일이 무엇인지 설명
 - 변수 설명
 - 각 단계의 맥락을 설명
 - 끝이 어디인지 확인
 - 긴 프로그램의 경우 procedures 가 어디서 끝나는 지 확인할 때

- 50 마리 생성

```
crt 50
```

- local hilltop에 올라오면 멈춤

```
to move
```

```
  if elevation >=
```

```
    [elevation] of max-one-of neighbors [elevation]
```

```
    [stop]
```

- corridor width를 계산해보자
 - 나비가 들른 patch의 수

```
patches-own  
[  
  used?  
]
```

Note boolean (true-false) 변수: 변수명 끝에 ?

- 출발점과 도착점의 평균 거리

```
turtles-own  
[  
  start-patch  
]
```

- 새로 만든 변수를 초기화 해주어야 함

```
ask patches
```

```
[
```

```
...
```

```
  set used? false
```

```
]
```

```
crt 50
```

```
[
```

```
...
```

```
  set start-patch patch-here
```

```
]
```

Note initializing variables: 새로운 변수는 다른 값을 지정하기 전에는 0에서 시작. 프로그램 작동에는 문제를 일으키지 않지만, 분석에서는 문제가 될 수 있는 데, 알기도 찾기도 어려운 error → 값을 주고 시작하는 습관을 들이자.

- 나비가 지나간 patch를 계산해야 함

```
to move
```

```
...
```

```
  set used? true
```

```
  end
```

- 이제 corridor width 를 계산해야 함

- local 변수: final-corridor-width,

- 보고문 corridor-width

```
to go
```

```
...
```

```
  if ticks >= 1000 [stop]
```

```
  let final-corridor-width corridor-width
```

```
end
```

```
...
```

```
to-report corridor-width
```

```
end
```

```
to-report corridor-width
  let num-of-visited-patches count patches
  with [used? = true]

  ask turtles
  [
    set the-distance distance start-patch
  ]

  set mean-of-the-distance mean [the-distance]
  of turtles
  report num-of-visited-patches / mean-of-the-distance
end
```

```
globals  
[  
  ; q  
  mean-of-the-distance  
]  
  
turtles-own  
[  
  start-patch  
  the-distance  
]
```

- 결과 관찰하기

- 그래프 그리기

```
to go
  ask turtles [move]
  plot corridor-width
  ...
end
```

- Interface 탭에 그래프 창 그리고
 - 그래프 이름은 Corridor Width
 - Update commands의 내용은 모두 삭제

- csv 파일로 내보내기

```
to go
  ...
  export-plot "Corridor width"
  (word "Corridor-output-for-q-" q ".csv")
end
```

- UTM 같은 좌표 시스템을 바로 NetLogo에서 사용할 수 없음
→ 전환 필요. 관련 소프트웨어에서 알아서
 - 가로로 한 줄에, grid point or cell 좌표 정보, 공간 특성 정보가 들어가도록 작성
 - NetLogo의 patch 하나의 크기(1×1)에 맞춰 실제 공간의 scale을 보정 해주어야 함
- NetLogo 모델에서 공간 설정
 - Settings 창 뜨는
 - 명령어 `resize-world`

- 공간 파일을 불러옵니다.

- 파일 버전 변경
- 불러올 공간 파일을 같은 폴더에 둬. 아니면 user-file 명령어 사용

```
to setup
```

```
...
```

```
  file-open "ElevationData.txt"
```

```
  while [not file-at-end?]
```

```
  [
```

```
    let next-x file-read
```

```
    let next-y file-read
```

```
    let next-elevation file-read
```

```
    ask patch next-x next-y [set elevation next-elevation]
```

```
  ]
```

```
  file-close
```

```
end
```

- max, min을 써서 최고 높이와 최저 높이에 따라 색을 변화

```
to setup
```

```
...
```

```
file-close
```

```
ask patches
```

```
[
```

```
  set pcolor scale-color green elevation
```

```
    min [elevation] of patches
```

```
    max [elevation] of patches
```

```
  set used? false
```





```
]
```

```
end
```

BehaviorSpace

- 서로 다른 시나리오가 반복되는 실험(experiments)을 해야 함
 - 모델과 변수/입력 자료/초기 조건의 한 집합이 하나의 시나리오
 - 만약 확률 과정이 없다면, 하나의 시나리오를 여러 번 반복하더라도 동일한 결과가 나올 것
 - 반복 실험은 모델에서 확률 과정만 변화하고 다른 요소는 그대로 두고 실행하는 것
 - 무작위로 생성되는 수가 바뀌거나
 - 입력 자료를 바꾸거나
 - 초기 값을 바꾸는 것도 다른 시나리오가 될 수 있음

- BehaviorSpace가 우리를 구원하리라.
 - NetLogo User Manual → Features → BehaviorSpace Guide.
 - 전역 변수 값을 바꿔서 시나리오를 생성
 - 각 시나리오를 반복 시행 (repetitions)
 - 각 시행의 결과를 모아서 하나의 파일로 작성
- 우리의 경우, q

-  Bonabeau, E. (2002).
Predicting the unpredictable.
Harvard Business Review.
-  Page, S. E. (2008).
Agent-based models.
In Durlauf, S. N. and Blume, L. E., editors, *The New Palgrave Dictionary of Economics*. Palgrave Macmillan.
-  Rand, W. and Rust, R. T. (2011).
Agent-based modeling in marketing: Guidelines for rigor.
International Journal of Research in Marketing, 28:181–193.
-  Starfield, A. M., Smith, K. A., and Bleoloch, A. L. (1990).
How to model it: Problem solving for the computer age.
McGraw-Hill.