エージェントベースシミュレーションソフトNetLogo

○吉田孝志（日本電気株式会社）　倉橋節也（筑波大学大学院）

概要 社会シミュレーションに適したツールである NetLogo について紹介する．NetLogo はオープンソースで開発されている汎用エージェントベースシミュレーションソフトであり，平易な言語でモデルを記述し実行することができる．またインターネット上では NetLogo で記述されたさまざまなモデルが公開されており，これを利用することもできる．開発の中心は米国であるが，マニュアル類の日本語化を著者らが進めている．

キーワード: エージェントベースシミュレーション，プログラミング言語，オープンソース

1　はじめに

社会科学における数理モデルを自ら設計し，コンピュータ上に実装してシミュレーションを実行させたいならば，モデルの挙動を何らかのプログラミング言語でコーディングする必要がある．しかし Java などの汎用的なプログラミング言語を用いようとすると，まず習得までに時間がかかるし，習得できたとしてもモデルを動かすまでには，オブジェクトの設計はどうするか，可視化の方法はどうするかといった，モデリングとは直接関係のない各種の要素に悩まされることになる．

そのため世界の社会シミュレーション研究者の間では，モデリングとシミュレーションの機能に特化したシミュレーションソフトが利用されている．本稿では，これらの中でも近年主要なツールとして広く利用されている汎用エージェントベースシミュレーションソフトである NetLogo を取り上げ，その特徴や活用例を紹介する．

NetLogo の開発は米国を中心として行われており，マニュアル類も大部分は英語であるため日本語話者には使いづらい面もあるが，さまざまなサンプルモデルをフリーで入手できるというメリットがある．また著者らはマニュアル類の日本語化を進めており，初学者にとってのハードルは低くなってきている．

2　NetLogo の概要

NetLogo は，主に社会科学系や生態学系の研究者に利用されているエージェントベースシミュレーションソフトである．エージェントベースモデルの記述に配慮した NetLogo 言語を用いてモデルをコーディングしシミュレーションを実行させることができる．したがってたとえば Social Force モデル [1] のような特定のモデルしか組み込まれていないシミュレーションソフトとは異なり，自分で任意のモデルを設計して動作させることが可能である．一方で Java などの汎用的なプログラミング言語とも異なり，エージェントの概念や，シミュレーションの実行管理，グラフィカルインターフェースといった枠組みがはじめから提供されているため，これらの枠組みの検討に煩わされることなく，モデルそのものの内容の検討に集中することができる．

NetLogo には各種のモデルがバンドルされているほか，世界の研究者が NetLogo を用いて開発したモデルがインターネット上で数多く公開されている．社会科学における著名な数理モデルについては，わざわざスクラッチからコーディングせずとも，公開されているモデルを借用してきてパラメータの値を操作し，モデルの挙動がどう変化するか観察してみるといったことも手早く試すことができる．

NetLogo はオープンソースソフトウェア (OSS) でありフリーで利用できることも特徴の 1 つである．1990 年代に Uri Wilensky が最初のバージョンを公表し，以降は GitHub[2] に開発の場を移し，現在に至るまで継続して開発が進められている．2016 年 12 月にはバージョン 6 がリリースされている．

英語では NetLogo の教科書 [3] や，NetLogo を用いたシミュレーションによって社会現象の説明を試みた書籍 [4] が出版されている．日本語でのドキュメントは少ないが，いくつか論文 [5] が出ているほか，マニュアル類の日本語化 [6] を著者らが進めている．また NetLogo のインターフェースが長らく英語であったことも日本語話者にとっては障壁の 1 つであったが，バージョン 6 からは著者らが作成した日本語メニューがバンドルされるようになった．

3　NetLogo の機能

3.1　インストール

NetLogo は Java 仮想マシン上で動作する．Java の知識そのものは不要である．Java 仮想マシン上で動作するプログラミング言語としては Jython や JRuby 等があるが，これらと同様の仕組みといえる．Java が動くならばプラットフォームには依存しないため，Mac, Windows, Linux などの主要なプラットフォームで実行可能であり，またいずれかのプラットフォームで作成したモデルを他で動かすこともできる．

NetLogo は公式サイト [7] からダウンロードしインストールする．標準のダウンロード版には Java がバンドルされているため別途 Java をインストールする必要はない．なお NetLogo では高速化のため JRE ではなく JDK を使用している．

3.2　画面

NetLogo を起動すると,「インターフェース」「情報」「コード」の 3 つのタブが表示される（Fig. 1）．インターフェースタブには，モデルを操作したりパラメータを設定したりするためのボタン，スライダー，スイッチなどの機能や，モデルの実行結果を時系列グラフとして表示するためのプロットなどの機能が配置される．

インターフェースタブにはまたエージェントを表示するためのワールドウィンドウが配置される．ここではエージェントの挙動を視覚的に観察できるほか，特定のエージェントをクリックすればそのエージェントが持つ変数の値を表示することができる．これによりモデルの細部の動作を確認でき，デバッギングも容易になる．

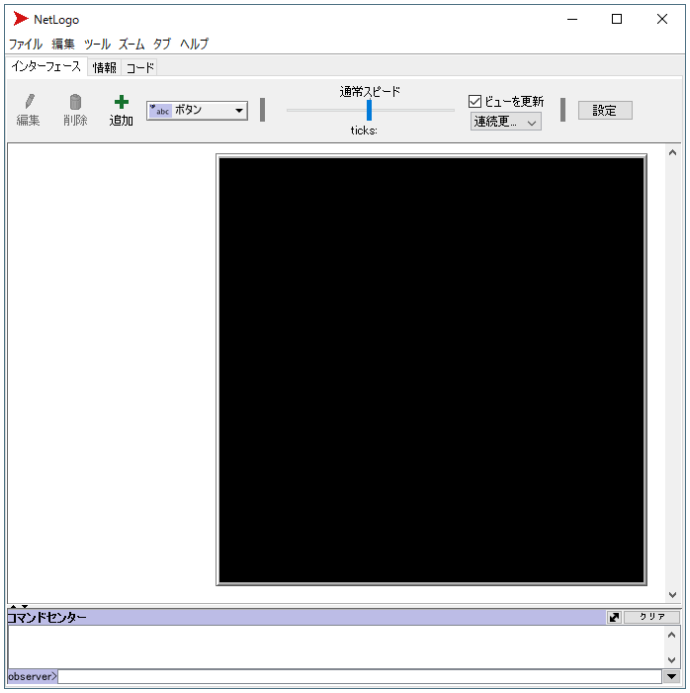

Fig. 1: NetLogo の画面

インターフェースタブの下部には「コマンドセンター」が配置されている．ここでは新たにエージェントを生成するなどのコマンドを指示することができる．

情報タブには，軽量マークアップ言語のひとつである Markdown 言語を用いて，モデルについての説明等を記述する．

コードタブにはソースコードを記述する．コードタブはモデルの開発環境を兼ねており，検索・置換やシンタックスハイライトの機能を有している．ソースコードは`__include` コマンドを用いれば別のテキストファイルに記述することも可能であり，使い慣れたエディターでコーディングを行うこともできる．ただしモデルの実行にはインターフェースタブで定義される情報 (座標系の設定や一部のグローバル変数など) が必要であり，ソースコードだけではモデルを実行することはできない．

3.3 補助的機能と拡張機能

NetLogo は各種の補助的機能を有している．さらにユーザーが拡張機能を自分で開発して付加できる機能もあり，有志の開発者が拡張機能を開発し GitHub などで公開している．拡張機能の中でも重要なものは公式のダウンロード版にバンドルされている．

補助的機能の中で使用頻度が高いものとしては動画作成機能があり，シミュレーションの進行を録画してプレゼンテーションなどで活用することができる．背景として地図画像などの画像ファイルをインポートして表示することも可能である．

BehaviorSpace はパラメータスイープ機能である．モデルのパラメータを連続的あるいは任意に変化させ，実行結果を記録することができる．マルチコア CPU にも対応しており，シミュレーションの並列実行も自動的に行ってくれるため，モデルの検証や多数の実験を繰り返す場合に有効である．

HubNet は複数の端末で同時利用できる参加型シミュレーション機能である．たとえばゲーミングの機能を有するモデルを用意し，授業などで学生にパラメータの一部を操作させ，教員がレフェリーとなってその結果を判定するといった使い方ができる．

拡張機能を使うには `extensions` コマンドを用いる．代表的な拡張機能には Watts-Strogatz モデル [8] などのネットワーク構造を作成したり中心性などの指標を求めることができる Network Extension や，ESRI 形式の GIS データを取り込めるようにした GIS Extension などがある．

4 NetLogo のモデル

4.1 モデルの構成要素

NetLogo を用いて設計されるモデルは「ワールド」と「エージェント」から構成される．ワールドはエージェントが動き回る 2 次元の平面である．NetLogo にバンドルされている NetLogo 3D では 3 次元空間のワールドを扱うこともできる．

エージェントには「タートル」「パッチ」「リンク」「オブザーバー」の 4 種類がある．タートルはワールドを動き回ることができるエージェントである．パッチはワールドにおけるタイル状の座標面であり，動くことはできないが他のエージェントと相互作用する．リンクは 2 つのタートルを結びつけるエージェントである．オブザーバーはモデルやエージェントを外部から操作する存在であり，コード上では直接には表れない概念的な存在である．タートルとリンクには「品種 (breed)」を定義し，品種ごとに異なる性格を与えることもできる．

モデルを実行するとワールド上にパッチが生成される．ワールドのサイズを 100×100 に設定すれば，生成されるパッチは 10,000 個である．オブザーバー，パッチ，タートルは他のタートルを作ることができ，オブザーバーとタートルはまたリンクを作ることができる．これらのエージェントはモデル上の時刻に相当する「ティック（tick)」の経過とともに自ら変化したり，他のエー

ジェントと相互作用したりする．タートルとリンクは死んで消滅する場合もある．モデルの内容や計算機の性能にもよるが，同時に数千個から数万個のタートルを動作させることができる．

4.2 NetLogo 言語

NetLogo ではモデルの記述に独自のプログラミング言語である NetLogo 言語を用いる．NetLogo 言語はインタープリター型の設計による手続き指向言語であり，1960 年代に教育用に開発された LOGO 言語を起源としている．Java などのオブジェクト指向言語に慣れた人には多少違和感があるが，プログラミングになじみの薄い学生等にとっては取っ付きやすいようである．一方でエージェントベースモデルを記述するための各種の機能が用意されており，一般に Java などに比べずっと少ない行数でのコーディングが可能である．手続き指向言語であることから比較的小規模なモデルのコーディングに適していると言えるが，複雑なモデルの記述も可能である．

NetLogo で 1 行だけのコマンドを実行するには，コマンドセンターにたとえば次のように入力する．

```
print "Hello, World!"
```

複数行にわたるコマンドを実行するにはコードタブでプロシージャを定義する．プロシージャは戻り値がない場合は to で始まり end で終わるコマンドプロシージャ，戻り値がある場合は to-report で始まり間に report を含み end で終わるレポータープロシージャとする．下記は NetLogo のコードの例である．

```
globals [SPEED]
turtles-own [Rotation]

to setup
  ca
  set SPEED 1
  let degree 10
  crt NUMBER [
    set Rotation random degree
    move-to one-of patches
  ]
  reset-ticks
end

to go
  ask turtles [
    fd SPEED
    rt Rotation
  ]
  tick
end
```

まず冒頭の globals でグローバル変数 SPEED を宣言し，turtles-own でタートル変数 Rotation を宣言している．タートルは他にも座標，向き，形状などの変数を持っているが，これらは NetLogo の枠組みの中ではじめから提供されているため宣言は不要である．またグローバル変数はインターフェースタブでスライダーなどを作成するとその中で自動的に宣言される．上記のコードでは NUMBER という変数がこれにあたる．なお変数の型宣言は不要である．また変数名の大文字と小文字とは区別されない．

setup プロシージャではモデルの初期設定を定義している．まず ca ですべての変数や表示をクリア (clear all) し，set SPEED 1 でグローバル変数 SPEED の値を 1 に設定し，let degree 10 でプロシージャ内でのみ有効なローカル変数 degree を宣言するとともに値を 10 に設定している．次に crt NUMBER でタートルを NUMBER 個生成し，同時に生成したタートルに [] 内のコマンドを実行させている．その内容は，set Rotation random degree でタートル変数 Rotation の値を 0 から 9 までのランダムな整数に設定し，move-to one-of patches でランダムに選んだいずれかのパッチの位置に移動するものである．最後に reset-ticks でティックカウンタを初期化している．

go プロシージャではモデルのステップごとの挙動を定義している．まず ask turtles ですべてのタートルに対して [] 内のコマンドを実行させている．その内容は，fd SPEED で前方へ SPEED の距離を進み，rt Rotation で右向きに Rotation 度回転するものである．なお SPEED の値は全タートルで共通であり，Rotation の値は各タートルでそれぞれである．最後に tick でティックを 1 進めている．

なお NetLogo のモデルでは setup と go というプロシージャ名を用いるのが慣例となっているが，必ずしもこれに従う必要はない．

このようにして定義したプロシージャはどこからでも呼び出すことができる．たとえばコマンドセンターで setup と入力すれば setup プロシージャが実行され，次に repeat 100 [go] と入力すれば go プロシージャが 100 回実行される．多くのモデルではインターフェースタブ内に「setup」ボタンと「go」ボタンを用意してプロシージャを呼び出せるようにしている．また「go」ボタンに「フォーエバー」オプションを指定して，同じプロシージャが繰り返し実行されるよう設定している場合も多い．

ask turtles [] はすべてのタートルに [] 内のコマンドを実行させるコマンドであるが，NetLogo 言語の特徴の 1 つとして，エージェントが動作する順序は自動的にランダム化される．エージェントベースシミュレーションにおいては，エージェントが動作する順序に意味がある場合 (たとえば，先にタートルがいるパッチには後から入れない，など) に，順序がランダム化されていないと意図しない結果につながる可能性がある．このようなときに Java 等ではエージェントの動作順序を常に意識する必要があるが，NetLogo を使えばその心配も不要である．もちろん，常にエージェントの 0 番から同じ順序で動作させたいといった場合は，そのようにコーディングすることも可能である．

4.3 モデルの例

NetLogo のモデルの例として Schelling の分居モデル [9] を紹介する．このモデルは，異なる人種が隣りあって住むことに対する寛容性の強弱が，人種によって住む地域が分かれる状況を生じさせているとするものである．住民を模したエージェントが，周囲に住む他の

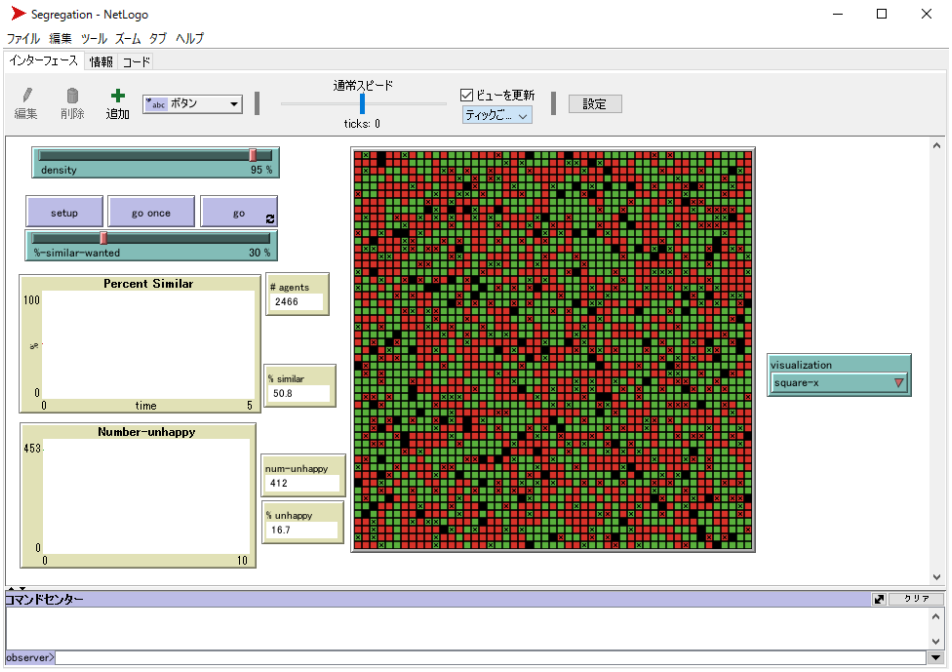

Fig. 2: NetLogo にバンドルされている Schelling の分居モデル (Uri Wilensky, CC BY-NC-SA)

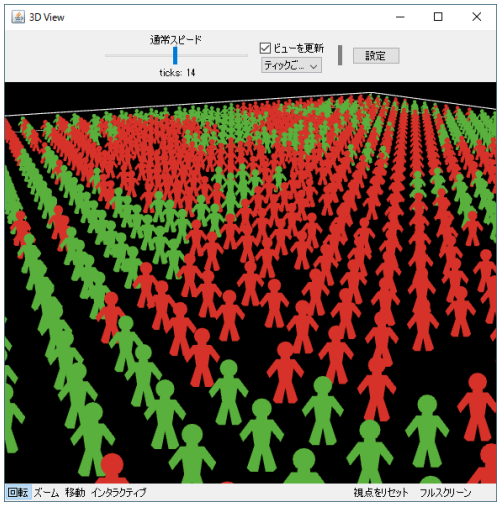

Fig. 3: Schelling の分居モデルの 3D 表示

エージェントの一定比率以上が自分と同じ人種であれば満足して定住し，それ以下であれば転居するものとする．シミュレーションを行ってみると，「周囲の 60%が同じ人種なら満足する」という程度の寛容性であったとしても，時間の経過とともにはっきりとした分断が生じていく．一方で「周囲の 80%が同じでないと満足しない」というように寛容性が低くなると，逆に安定しなくなるという現象が起きる．

Schelling の分居モデルは NetLogo にバンドルされており，「ファイル」メニューから「モデルライブラリ」を選択し，「Sample Models」「Social Science」「Segregation」と選択すると起動される（Fig. 2）．モデルを起動させたら，インターフェースタブの「density」スライドバーと「%-similar-wanted」スライドバーでパラメータを調整し，「setup」ボタンを押してエージェントを初期化する．次に「go once」ボタンを押すとエージェントが 1 ステップだけ動作し，「go」ボタンを押すと状態が安定するまでステップごとの動作を繰り返す．

次にモデルをカスタマイズしてみる．コードタブを開き，次のように書かれている個所を削除する．

```
;; add visualization here
if visualization = "old" [ set shape
  "default" ]
if visualization = "square-x" [
  ifelse happy? [ set shape "square" ]
  [ set shape "square-x" ]
]
```

代わりに同じ個所に次のように記載する．

```
set shape "person"
```

インターフェースタブに戻り「setup」ボタンを押すとタートルが人間の形になる．次にワールドウィンドウをクリックして「Switch to 3D View」を選択すると Fig. 3 のような 3D 表示が別ウィンドウで表示される．

5 NetLogo の利用例

NetLogo には Schelling の分居モデルの他にも，社会科学系，生態学系などを中心とするサンプルモデルや，授業で使えるよう設計したモデルなど，約 350 個のモデルがバンドルされている．さらに Uri Wilensky らは NetLogo モデルの共有サイトである Modeling Commons[10] を運営している．このサイトにはユーザーがアップロードしたモデルが 2017 年 2 月現在 1,700 個以上掲載されており，検索やコメント記載の機能も有している．他にエージェントベースモデルの研究者のコミュニティである OpenABM[11] は，NetLogo を含む各種のプラットフォームを用いて開発された約 400 個のモデルを掲載している．

NetLogo は学会等でも広く利用されている．European Social Simulation Association の年次学術大会である Social Simulation Conference 2014 (ESSA/SSC 2014) での実例を見ると，全部で 80 本の Proceedings

のうち，NetLogo に言及があったものは 20 本，うち「モデリングに NetLogo を使った」と明記されていたものは 18 本あった．なお NetLogo に類似するエージェントベースシミュレーションソフトである RePast[12] の場合は，言及があったもの 8 本，モデリングに使ったもの 5 本という状況であった．

6 NetLogo の欠点

NetLogo にはいくつか欠点もある．まずエージェントの可視化などの機能に計算リソースを割り当てているため，計算機の性能にもよるが，エージェント数を増やすと実行速度が遅くなる．特にワールドのサイズを広げた場合にその影響が顕著となる．これは，たとえばワールドのサイズを $1,000 \times 1,000$ とすると 100 万個のパッチが生成されるためである．

NetLogo 言語については，手続き指向言語に共通の欠点として，コードの行数が増えてくると次第に難しい面が出てくる．とりわけグローバル変数を増やしすぎるとコードのスパゲティ化を招くことが多く，ローカル変数やレポータープロシージャを活用して変数のスコープを限定する等の設計が必要となる．

また NetLogo 言語は独自言語であるため他の汎用言語で書かれたシステムとの連携が難しい．Java からは Controlling API の機能を使って NetLogo をコントロールすることが可能であるが，たとえば機械学習系の Python のライブラリと連携させようとすると面倒である．この点で相対的に優れていると思われるのは Python で書かれている国産の S4 (エス・クワトロ)[13] というシミュレーションソフトである．

NetLogo は OSS であるため，バージョンアップの際に後方互換性が必ずしも保障されない．最近のバージョンアップではこの点は配慮されているが，過去にはバージョンアップの際にコードの手直しが必要になる場合もあった．また OSS であるためサポートもない．この点については著者らができるだけ利用者の手助けをしたいと考えている．

7 まとめ

近年世界で広く利用されているエージェントベースシミュレーションソフトである NetLogo について紹介した．NetLogo は社会科学における数理モデルを平易なプログラミング言語でコーディングでき，また各種の補助的機能や拡張機能を有している．OSS であるためフリーで利用でき，さらにインターネット上で公開されている数多くのサンプルモデルを利用することもできる．日本語でのドキュメントは少ないが，著者らがマニュアル類の日本語化を進めており，日本でも NetLogo の利用が広がることを願っている．

参考文献

1) Helbing, D., and Molnar, P.: Social force model for pedestrian dynamics, *Physical review E*, **51**-5, 4282 (1995)
2) `https://github.com/NetLogo/`
3) Wilensky, U., and Rand, W.: *An Introduction to Agent-Based Modeling: Modeling Natural, Social and Engineered Complex Systems with NetLogo* (2015)
4) Epstein, J. M.: *Agent Zero: Toward Neurocognitive Foundations for Generative Social Science* (2014)
5) 倉橋, 田中, 小林:社会科学におけるエージェントモデリング環境― NetLogo, Repast Simphony, Repast for High Performance Computing ―，人工知能，**30**-4，460/467 (2015)
6) `http://www2.gssm.otsuka.tsukuba.ac.jp/staff/kurahasi/netlogo.html`
7) `http://ccl.northwestern.edu/netlogo/`
8) Watts, D. J., and Strogatz, S. H.: Collective dynamics of 'small-world' networks, *Nature*, **393**-6684, pp.440442 (1998)
9) Schelling, T. C.: Dynamic models of segregation, *Journal of mathematical sociology* **1**-2, pp.143-186 (1971)
10) `http://modelingcommons.org/`
11) `https://www.openabm.org/`
12) `https://repast.github.io/`
13) `https://www.msi.co.jp/s4/`