LA CASSETTA DEGLI ATTREZZI

STRUMENTI PER LE SCIENZE UMANE

Simone Gabbriellini

SIMULARE MECCANISMI SOCIALI CON NETLOGO

Una introduzione

FrancoAngeli

La cassetta degli attrezzi. Strumenti per le scienze umane

La collana, rivolta a ricercatori accademici e professionisti, studiosi, studenti, e operatori del variegato mondo della ricerca empirica nelle scienze umane, si colloca sul versante dell'alta divulgazione e intende offrire strumenti di riflessione e di intervento per la ricerca.
Obiettivo è consolidare le discipline umane presentando gli strumenti di ricerca empirica, sia di raccolta sia di analisi dei dati, in modo intellegibile e metodologicamente critico così da consentirne l'applicazione proficua rispetto a definiti obiettivi cognitivi.
I testi sono scritti da professionisti della ricerca che, attingendo alla personale esperienza maturata in anni di attività, offrono ai lettori strumenti concettuali e tecnici immediatamente applicabili nella propria attività di ricerca.
Tutti i volumi pubblicati sono sottoposti a referaggio.

Simone Gabbriellini

SIMULARE MECCANISMI SOCIALI CON NETLOGO

Una introduzione

La cassetta degli attrezzi
Strumenti per le scienze umane/6

FrancoAngeli

Progetto grafico di copertina di Maria Teresa Pizzetti

119. *La cassetta degli attrezzi. Strumenti per le scienze umane*

Volumi pubblicati:

1. Giovanni Di Franco, *L'analisi dei dati con SPSS. Guida alla programmazione e alla sintassi*
2. Silvia Cataldi, *Come si analizzano i focus group*
3. Federica Pintaldi, *Come si analizzano i dati territoriali*
4. Giovanni Di Franco, *Il campionamento nelle scienze umane: dalla teoria alla pratica*
5. Lucia Coppola, *NVivo: un programma per l'analisi qualitativa*
6. Simone Gabbriellini, *Simulare meccanismi sociali con NetLogo. Una introduzione*

Volumi in preparazione:

7. Alberto Marradi, *Come evitare gli errori tipici in un questionario*

Indice

1. Introduzione

1.1 Modelli ad agenti nelle scienze umane

L'obiettivo del testo è presentare ai ricercatori delle scienze umane l'uso dei modelli basati su agenti (Agent-Based Models, d'ora in poi ABM) come metodo per formalizzare, in termini computazionali, modelli teorici di meccanismi sociali, testarne la validità interna ed esaminare le conseguenze esplicite e implicite da essi derivabili.

Risolvendo un modello computazionale, le simulazioni con ABM permettono di condurre complessi esperimenti mentali. Il loro valore aggiunto è di essere un laboratorio virtuale in cui poter analizzare il meccanismo ipotizzato e ricostruire il difficile percorso micro-macro: non potremmo mai compiere esperimenti mentali così complessi senza un computer. In questo senso, le simulazioni ad agenti sono state definite "aid-intuition" (Axelrod 2003) o "lenti d'ingrandimento" (Terna 2009). Per quanto il primo scopo sia utilizzarle per "prevedere cosa succederà", in realtà l'obiettivo principale di una simulazione con ABM resta quello di dettagliare dei meccanismi per "capire cosa è successo", contribuendo così a sviluppare una teoria (Gilbert, Terna 1999; Epstein 2008).

Programmare i comportamenti degli agenti in modo da essere eseguiti da un computer impone la costruzione di una teoria, per il semplice motivo che dobbiamo dire al computer cosa fare, e dobbiamo farlo in modo sufficientemente dettagliato da essere computabile. Allo stesso tempo, la computabilità della versione informatica è garanzia dell'assenza di *black boxes* nella teoria, perché queste ultime non

potrebbero essere capite dal computer – e per quanto ne sappiamo, neppure dal ricercatore (Manzo 2007).

Per ogni ABM esiste anche un corrispettivo insieme di formule matematiche che lo descrivono, ma che non abbiamo bisogno di dettagliare (Epstein 2006; Leombruni, Richiardi 2005). In particolare, se le equazioni che descrivono un processo sociale:

- possono essere scritte *ex-ante* e risolte analiticamente, la simulazione ad agenti è meramente uno strumento di presentazione dei risultati;
- possono essere scritte *ex-ante* e risolte numericamente, la simulazione ad agenti è un'analisi alternativa alle simulazioni basate sul metodo Monte Carlo (cioè quella famiglia di metodi statistici utilizzati per produrre stime attraverso simulazioni numeriche, calcolando una serie di realizzazioni possibili del fenomeno in esame e cercando di esplorare in modo denso tutto lo spazio dei parametri del fenomeno);
- possono essere scritte *ex-ante* ma non possono essere completamente risolte, allora la simulazione ad agenti risulta essere uno strumento essenziale per far luce sulle dinamiche del sistema;
- non possono essere scritte *ex-ante*, allora la simulazione ad agenti risulta essere l'unico modo di trattare il sistema (Axtell 2000).

Far "girare" un modello significa istanziare un certo numero di agenti (una popolazione), lasciarli interagire, e monitorare quello che accade. In questo modo verifichiamo le conseguenze delle caratteristiche e delle abilità relazionali attribuite agli agenti nell'ambiente in cui operano.

Grazie agli ABM, possiamo studiare la non linearità dei comportamenti aggregati rispetto all'azione degli agenti: si ottengono così risultati (anche inattesi) sotto forma di emergenza di fenomeni, regolarità e strutture.

Il concetto di emergenza ha una lunga storia in sociologia (Sawyer 2005). La concezione classica di emergenza come di fenomeno per principio inesplicabile, cui doversi semplicemente rassegnare, non è compatibile con la modellazione ad agenti. Piuttosto, l'emergenza è

semplicemente un fenomeno che in questo momento non siamo in grado di spiegare e che, proprio per questo, diviene l'obiettivo della spiegazione. A tale fine, non solo le caratteristiche dell'attore (i suoi dati di attributo), ma anche le sue abilità, la sua capacità d'interazione con gli altri (i suoi dati relazionali) entrano a far parte della descrizione del modello.

Il legame a doppia via tra micro e macro livelli è la ragione per cui è evocato in letteratura un approccio debole all'individualismo metodologico, detto individualismo strutturale (Hedstrom, Swedberg 1998; Hedstrom 2006; Manzo 2007, Hedstrom, Bearman 2010), che porta spesso, si veda ad esempio Terna (2006) e Squazzoni (2009), a distinguere due tipi fondamentali di emergenza, una di primo e una di secondo livello.

Con emergenza di primo livello ci riferiamo allo studio del legame micro-macro, dove regole semplificate di comportamento degli attori a livello micro producono l'emergenza di *pattern* complessi a livello macro. Quello appena descritto è un meccanismo noto in sociologia con il nome di *effetti non intenzionali dell'azione*, il cui oggetto è l'oscillazione tra semplicità micro e complessità macro, tra azione individuale e conseguenze non intenzionali emergenti dall'interazione.

Con emergenza di secondo livello ci riferiamo invece allo studio di legami a doppia via tra comportamenti micro e l'emergenza di complessi *pattern* macro, attraverso circoli ricorsivi tra contesto ed azione. In questo caso, l'analisi degli effetti emergenti è accoppiata con l'analisi dei meccanismi macro-micro che influiscono sui processi decisionali dell'attore. Anche in questo caso, il meccanismo descritto è già noto e studiato in sociologia con il nome di *Coleman Boat* (Coleman 2005), illustrato in figura 1.1.

La costruzione di un ABM è sostanzialmente il prodotto di quattro passaggi:

- definizione degli agenti (quanti, di che tipo);
- definizione degli attributi degli agenti (dati di attributo);
- definizione delle regole che legano gli attributi degli agenti;
- definizione delle regole che legano gli agenti tra loro.

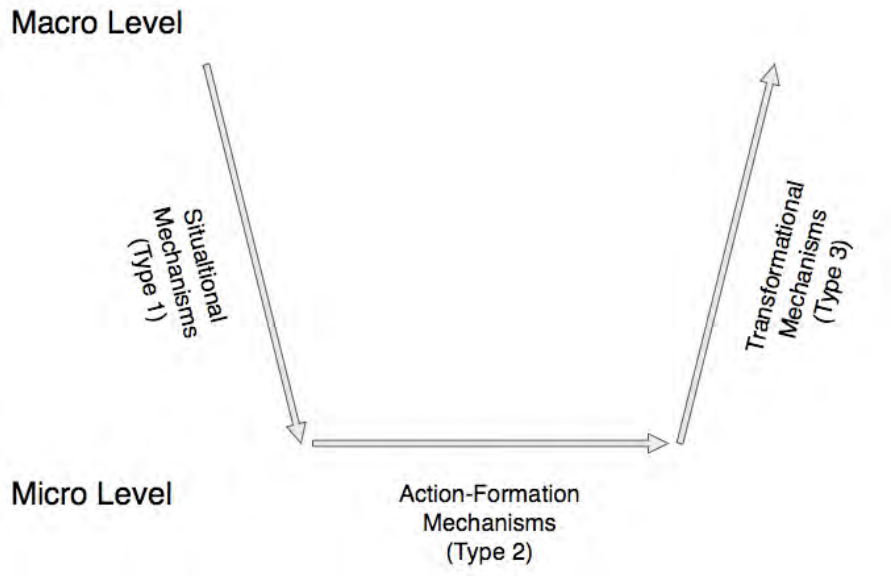

Figura 1.1 – Il Coleman's Boat nella versione proposta da Hedstrom, Swedberg (1998).

In ognuno dei passaggi è necessario ricorrere alle nostre capacità inventive, essenzialmente in base a: creatività, logica, matematica e statistica. Attraverso la simulazione dell'ABM, cioè avviando l'interazione degli agenti e facendo scorrere il tempo simulato, è possibile osservare le regolarità contenute nel meccanismo.

I modelli presi in esame in questo lavoro provengono da articoli pubblicati in riviste sociologiche e politologiche internazionali. Ciascuno presenta un problema, una teoria e un modello dei meccanismi sociali ipotizzati dall'autore. Per ogni modello, vengono ipotizzate delle caratteristiche e delle abilità possedute dai singoli attori sociali e analizzati i risultati a livello aggregato prodotti delle loro interazioni simulate.

Implementeremo i modelli trasformandoli in ABM capaci di riprodurre i risultati presentati nei rispettivi articoli.

Gli autori selezionati hanno differenti appartenenze teoriche, e l'esteso arco temporale coperto dai loro articoli fa sì che non tutti abbiano effettivamente usato modelli ad agenti nelle loro implementazioni, né che tutti presentino esplicitamente i propri modelli come

formalizzazioni di meccanismi sociali, o che siano presenti riferimenti alla sociologia analitica – in questo momento l'unico approccio sociologico che faccia seriamente e programmaticamente uso di ABM (Hedstrom 2006; Manzo 2010; Hedstrom, Bearman 2010). Tuttavia, la similarità nell'approccio metodologico e nell'utilizzo di modelli formali ci permette di tradurre agilmente il loro lavoro in moderni ABM e ricondurne il metodo a quella che Epstein (2006) ha definito *scienza sociale generativa* – per quanto l'idea di spiegazione generativa non sia una novità in ambito sociologico (Boudon 1979).

Alla base di un approccio generativo c'è la volontà di spiegare la genesi delle regolarità osservate, e non descriverne semplicemente l'andamento: dovremo riprendere i meccanismi sociali dettagliati negli articoli, riprodurli e osservare i risultati, verificando lo scostamento con quelli presentati dagli autori.

Tutti i modelli qui presentati assolvono esclusivamente la funzione che Macy e Willer (2002) identificano nello sviluppare una teoria e *controllarne la validità interna*. Occorre precisare inoltre che i modelli scelti rappresentano tutti esperimenti mentali svincolati da dati empirici. Sempre più spesso invece si richiede agli ABM un maggior legame con la realtà empirica che essi tentano di modellare. Sta dunque affermandosi l'uso di ABM calibrati empiricamente sia "in ingresso", utilizzando dati empirici a livello micro per impostare le condizioni iniziali di alcuni parametri della simulazione; che "in uscita", confrontando i risultati macro prodotti dalla simulazione con i dati raccolti empiricamente in una popolazione a livello aggregato (Hestrom 2006; Manzo 2007b).

Evidentemente, la complessità di tali modelli è maggiore rispetto a quelli qui analizzati e, per ragioni didattiche, abbiamo preferito non considerarli in questa sede.

Non abbiamo lo spazio per addentrarci a fondo in ogni articolo discusso, e lasciamo al lettore la possibilità di seguire i vari autori fino alle meno immediate implicazioni logiche dei modelli da loro proposti. Ci limiteremo a presentare i vari problemi che hanno ispirato le teorie e i modelli, a descrivere dettagliatamente questi ultimi, proponendone una traduzione in NetLogo, in modo che il lettore possa poi autonomamente condurre gli esperimenti proposti dagli autori degli articoli e valutare la similarità dei risultati ottenuti.

Va infine tenuto presente che le nostre traduzioni in NetLogo non devono essere ritenute in alcun modo le uniche possibili, né tantomeno le migliori.

Tutti i file dei modelli discussi nel volume possono essere scaricati all'indirizzo: http://www.digitaldust.it/Digitaldust/NetLogo.html.

Allo stesso indirizzo è possibile contattare l'autore per domande e/o eventuali problemi con il codice.

1.2 NetLogo

Lo scienziato sociale contemporaneo è abituato a usare il computer nel proprio lavoro, sia per la semplice scrittura sia per l'archiviazione dei dati così come per la loro analisi statistica o testuale. Potremmo definirlo un uso tradizionale. Per raggiungere il nostro obiettivo dovremo spingerci oltre, introducendo un uso avanzato del computer il cui obiettivo sia scrivere veri e propri programmi informatici.

Tale passaggio richiederà l'acquisizione di una certa familiarità con un linguaggio adatto a comunicare con il computer, cioè un linguaggio di programmazione.

Quando si parla di programmazione, la prima cosa che di solito viene in mente è che sia un'attività poco accessibile e forse anche un po' noiosa; una pratica logico-deduttiva che può rappresentare un *divertissement* solo per specialisti. Di sicuro, quello che non viene in mente è che la programmazione possa essere anche un'attività estremamente creativa e divertente.

Un po' come accade imparando una lingua straniera, una volta mandate a memoria le regole sintattiche e le parole chiave del linguaggio di programmazione, potremo dire al computer quello che vogliamo. E saremo anche liberi di dirlo prendendo la strada che preferiamo. Potremo essere garbati o arroganti, usare l'astuzia o la forza bruta, essere approssimativi oppure eleganti e raffinati. Certamente, una soluzione potrebbe rivelarsi meno efficiente di un'altra, ciononostante un linguaggio di programmazione ci rende liberi di immaginarle tutte – lasciando all'esperienza e alle necessità di ricerca la scelta di quella più adatta.

La creatività e la logica sono dunque gli ingredienti principali di questo approccio metodologico.

Inserire l'informatica nel bagaglio metodologico del ricercatore comporta delle difficoltà e pone una sfida notevole da affrontare, accrescendo la complessità del lavoro e introducendo potenzialmente nuove fonti di errore. Ciononostante, la complessità dei modelli depone a favore della computazione come metodo capace di far avanzare la teoria e, al tempo stesso, provare a superare la *querelle* metodologica tra qualitativo e quantitativo, tra la troppa vaghezza delle descrizioni verbali e la troppa semplicità dei modelli matematici (Fararo, Hummond 1998; Collins 1995).

I linguaggi di programmazione odierni sono strumenti molto potenti, astratti e complessi, generalmente basati sul concetto di 'oggetto' software; per questo, la programmazione che utilizza questi linguaggi è comunemente indicata come programmazione ad oggetti (OOP). Necessariamente, per produrre modelli professionali di una certa complessità e allo stesso tempo efficienti, il ricercatore dovrà acquisire familiarità con almeno uno dei linguaggi più avanzati in ambito di OOP, come C++, Objective-C, Java o Python. Dato però che l'introduzione alle basi della OOP esula dalle possibilità di questo lavoro, presenteremo qui una soluzione alternativa particolarmente adatta per avvicinarsi al mondo degli ABM, cioè NetLogo (Wilensky 1999).

NetLogo è un ambiente per la programmazione di ABM pensato per simulare fenomeni naturali e sociali. Esso riesce a catturare e riprodurre agevolmente l'aspetto dinamico di tali fenomeni, permettendoci di far interagire migliaia di agenti indipendenti, eterogenei e operanti in parallelo. In questo modo è possibile esplorare le connessioni tra i comportamenti a livello micro di tali agenti (eterogenei, autonomi, con informazioni limitate e capacità di interazione locale) e le conseguenze a livello macro che emergono da tali interazioni.

Il linguaggio di programmazione di NetLogo è un dialetto di Logo, un linguaggio procedurale nato per insegnare la programmazione ai bambini, dunque molto facile da apprendere. Ogni istruzione del linguaggio è rappresentata da una parola inglese, e spesso le sequenze di istruzioni somigliano a delle semplici frasi in quella lingua. Oltre alla semplicità del linguaggio, NetLogo offre una serie di stru-

menti indispensabili per osservare un modello che 'gira' nel computer, come monitor, grafici, campi di testo in cui inserire valori e così via.

Il punto di forza di NetLogo, quello che lo rende al tempo stesso facile da imparare e tuttavia sufficientemente potente da essere ritenuto un valido supporto anche dai ricercatori computazionali e analitici più esperti, è il cuore Java che lo anima: in pratica, gli strumenti di NetLogo e il dialetto Logo in cui scriviamo i modelli rappresentano uno *step* intermedio, per noi *user-friendly*, che NetLogo si occuperà di tradurre automaticamente in veloci e potenti modelli Java.

NetLogo è un software gratuito, sviluppato da Uri Wilensky presso il *Center for Connected Learning and Computer-Based Modeling* della NorthWestern University di Chicago ed è scaricabile all'indirizzo: http://ccl.northwestern.edu/netlogo/download.shtml.

Scarichiamo e installiamo la versione adatta al proprio sistema operativo (negli esempi che seguiranno, è stata usata la versione per Mac OSX). Appena apriamo il programma, ci troviamo di fronte una finestra con tre etichette, 'Interface', 'Information' e 'Procedures', come in figura 1.2.

Nella sezione 'Interface' è presente una finestra nera che rappresenta il World di NetLogo, cioè lo spazio all'interno del quale saranno creati gli agenti.

Nella sezione 'Information' potremo inserire descrizioni dettagliate del modello implementato.

Nella sezione 'Procedures', infine, inseriremo il codice del nostro modello. Cliccando in questa sezione, possiamo immediatamente renderci conto della libertà che il software offre: abbiamo a che fare con una pagina interamente vuota, in cui dovremo implementare da zero il nostro modello.

Il mondo di NetLogo è popolato di agenti. Gli agenti sono entità in grado di eseguire istruzioni. Ne esistono di quattro tipi:

- *turtles*, agenti mobili in grado di spostarsi nel World di NetLogo;
- *patches*, agenti quadrati che rappresentano porzioni fisse di 'terreno' del World di NetLogo;

- *links*, cioè turtles speciali che collegano una coppia di altre turtles. I link possono essere direzionati (*da* una turtle *a* un'altra turtle) oppure non direzionati (una turtle *con* un'altra turtle);
- *observer*, che può essere pensato come un agente in grado di osservare e controllare l'evoluzione del modello in ogni sua parte.

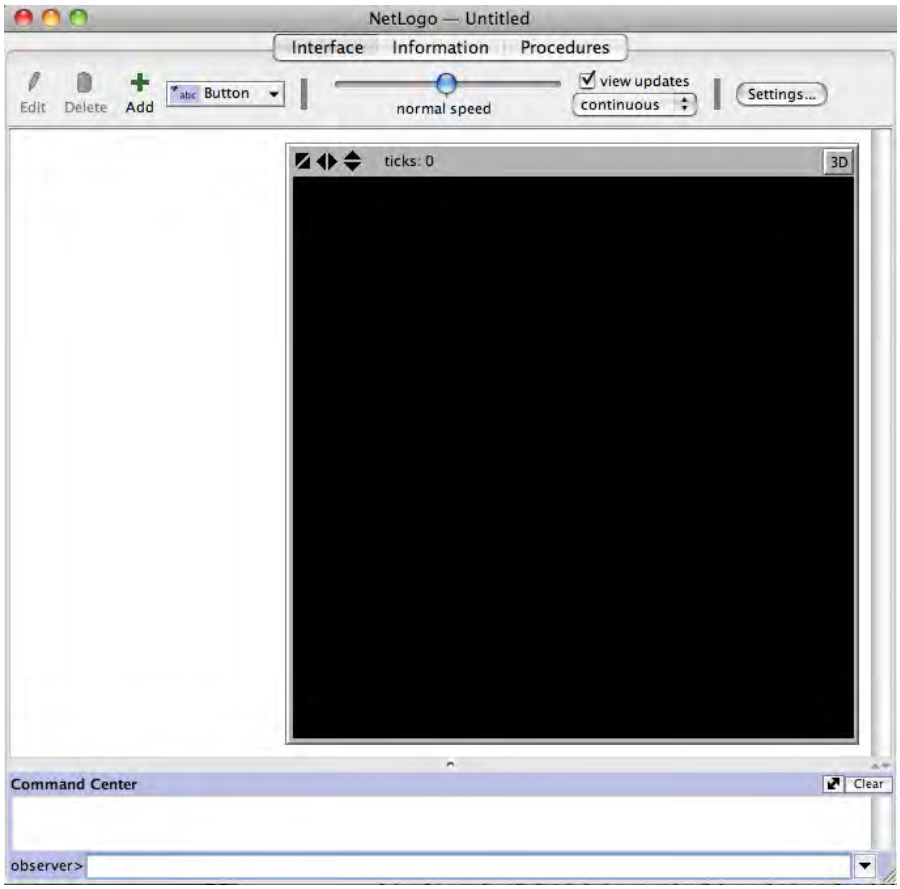

Figura 1.2 – La finestra di NetLogo all'apertura del programma

Le istruzioni dicono agli agenti cosa fare. In NetLogo esistono due tipi di istruzioni da far eseguire agli agenti: le *procedure*, implementate dall'utente, e le *primitive*, native del programma. Tipicamente, le procedure sono espressioni che racchiudono primitive, in una se-

quenza specificata dall'utente. Quando definiamo una procedura nella sezione Procedures, iniziamo il codice con la parola chiave *to*:

```
to nomeapiacere
```

e chiudiamo la procedura con la parola chiave *end*:

```
to nomeapiacere
end
```

Tra le due parole chiave possiamo inserire tutte le primitive che vogliamo, ad esempio, quella che pulisce il World di NetLogo, cancellando eventuali agenti, resettando eventuali plot e riportando a zero il tempo simulato, cioè *clear-all*, insieme alla primitiva per creare degli agenti, *create-turtles*:

```
to nomeapiacere    ;; è possibile inserire commenti con il punto e virgola
clear-all          ;; questa primitiva pulisce l'interfaccia di NetLogo
create-turtles 10  ;; questa invece crea 10 nuove turtle, o agenti
end                ; un punto e virgola è sufficiente.
```

All'interno di NetLogo adesso esiste una procedura in grado di creare 10 turtles (utilizzeremo questo nome per indicare gli agenti, dato che per definizione è quello usato in NetLogo). Come fare per eseguirla? Basta scrivere il suo nome nella sezione Observer, visibile dal pannello Interface. Immediatamente il World si popola di 10 turtles, disposte tutte al centro, una sopra l'altra.

Le istruzioni, sia procedure sia primitive, possono avere come obiettivo quello di calcolare e riportare un risultato (*reporter*) oppure di far eseguire un'azione alle turtles (*command*). Una procedura di report inizia sempre con la primitiva *to-report*:

```
to-report riportaValore
report count turtles     ;; la procedura riporta il conteggio (count) delle
end                      ;; turtles presenti nel World

to stampaValore          ;; questa procedura stampa (print) il valore riportato
print riportaValore      ;; dalla procedura riportaValore
end
```

Proviamo a scrivere stampaValore nell'Observer. Il risultato visualizzato nel Command Center sarà 10.

Nel caso invece di una procedura Command, l'obiettivo è quello di far eseguire delle azioni alle turtles. Vediamone un esempio:

```
to eseguiAzione
ask turtles [ forward 10 ]      ;; chiedi (ask) alle turtles
end                             ;; di andare avanti (forward) di 10 passi
```

Nella procedura eseguiAzione abbiamo specificato un command da far eseguire con la primitiva *ask*. Il command è sempre racchiuso tra parentesi quadre. Il compito della primitiva *ask* è fungere da ponte tra l'Observer e gli agenti, siano essi turtles, patches o links. Usando *ask* l'Observer (cioè il ricercatore) chiede agli agenti di eseguire dei comandi.

Le procedure possono anche ricevere un input quando vengono chiamate, come nel seguente caso:

```
to-report valoreDiInput [ numero ]
report numero + 1
end
```

Scriviamo nell'Observer:

```
print valoreDiInput 1232
```

Verrà visualizzato 1233 come risultato nel Command Center. Evidentemente, queste funzioni servono a illustrare concetti, piuttosto che a risolvere problemi. Non stupiamoci dunque della loro semplicità e della loro apparente insensatezza.

Un altro elemento chiave di NetLogo è rappresentato dal concetto di variabile. Una variabile è un contenitore per un valore, modificabile attraverso il comando *set*. Le variabili possono essere:

- *globali*: valori accessibili da tutte le turtles, e definibili sia attraverso l'uso di particolari elementi dell'interfaccia, come gli sliders, sia attraverso l'uso della parola chiave *globals* specificando il nome della variabile globale tra parentesi quadre;